



Grand Unified Debian



銀河系唯一のDebian専門誌

東京エリア/関西Debian勉強会



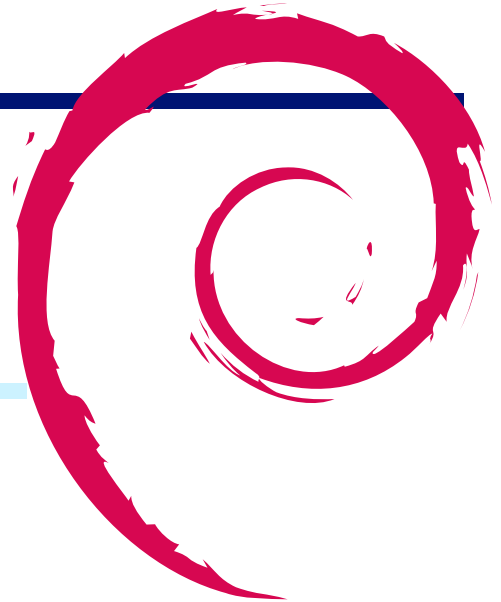
あんどきゅめんでっど でびあん 2017年夏号 2017年8月11日 初版発行

会 強 勉 的 ア ー ビ ュ

			について	17
	目次	6	Debbugs とのつきあいかた:SOAP 編	29
1	Introduction	2		
2	Debian Updates	3	Debian Continuous Integration	35
3	debhepler 10 の新機能を確認して みよう	8	challenge: convert debian-policy doc from docbook to Sphinx	40
4	qmake な Qt アプリの deb を作る うとして試行錯誤した話	12	2016 年の振り返りと 2017 年の企画	44
5	Debian パッケージのカスタマイズ	11	Debian Trivia Quiz	49
			Debian Trivia Quiz 問題回答	52

1 Introduction

DebianJP



1.1 東京エリア Debian 勉強会

Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか？

Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での「開発に関する情報」を整理してまとめ、アップデートする。
- 場の提供。
 - 普段ばらばらな場所にいる人々が face-to-face で出会う場を提供する。
 - Debian のためになることを語る場を提供する。
 - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりとするスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、「場」としての空間を提供するのが目的です。

1.2 関西 Debian 勉強会

関西 Debian 勉強会は Debian GNU/Linux のさまざまなトピック (新しいパッケージ、Debian 特有の機能の仕組、Debian 界隈で起こった出来事、などなど) について話し合う会です。

目的として次の三つを考えています。

- メーリングリストや掲示板ではなく、直接顔を合わせる事での情報交換の促進
- 定期的に集まれる場所
- 資料の作成

それでは、楽しい一時をお楽しみ下さい。

2 Debian Updates

杉本 典充



2.1 Agenda

- Debian 9 情報
- Debian Updates
- Debian とは?
- 今後のイベント

2.2 Debian 9 について

Debian 9.0 (コードネーム : Stretch) を 2017-06-17 にリリースした。

このリリースは、Debian Project の創始者 Ian Murdock 氏に捧げるリリースになっている。

サポートアーキテクチャ

- i386 アーキテクチャのサポート CPU を i686 以降に変更
- サポートされるアーキテクチャ
amd64, i386, armel, armhf, arm64, mips, mipsel, mips64el, ppc64el, s390x
- サポートから外れたアーキテクチャ
powerpc

<https://www.debian.org/releases/stable/amd64/release-notes/ch-whats-new.ja.html>

テーマ

- Juliette Taka Belin さんが作成した Soft waves を採用

ソフトウェア

- Linux カーネルは 4.9
- ツールチェイン (GCC 6.3.0, binutils 2.28, glibc 2.24), LLVM 3.7.1, 3.8.1, 3.9.1
- Perl 5.24.1, Python 2.7.13/3.5.3, Ruby 2.3.3, PHP 7.0.19, Go 1.7.4, OpenJDK 8
- GNOME 3.22, KDE 5.8, Xfce 4.12.3, lxde 0.99.0, lxqt 0.11.1
- MariaDB 10.1.23, PostgreSQL 9.6.3, sqlite 3.15
- OpenSSL 1.1.0, GnuPG 2.1.18/1.4.21
- クロスコンパイラがデフォルトでサポート
- etc..

パッケージの変更点

- iproute2 が推奨、net-tools は非推奨 (例 : ifconfig、arp、netstat、route)
- firefox、thunderbird という名称で提供
- mysql パッケージは提供されず、mariadb パッケージのみを提供
 - jessie からアップグレードする場合は、自動で mariadb パッケージに置き換えられる

- データベースは自動変換されるが、元に戻せないこと、失敗することもありうることを想定し、データ保全是各自の責任で実施すること

- Xorg サーバは root 権限でなくユーザ権限で動作することが可能

セキュリティ関係

- ウェブブラウザはセキュリティ更新が提供される Firefox および Chromium の利用を推奨
- Firefox 及び Thunderbird は、ESR 版のセキュリティ更新を提供
- libv8-3.14、nodejs、node-*はセキュリティ更新が提供されない
- OpenSSL において 3DES、RC4 暗号は TLS/SSL 通信には利用できない

互換性

- ネットワークインタフェース名が enp1s1 (ethernet)、wlp3s0 (wlan) のように変更
 - ただし、Debian 8 Jessie からアップグレードした場合は、eth0、wlan0 といった昔の命名規則で据え置き
- OpenSSH は標準で旧式の暗号と SSH1 プロトコルが無効
 - 古い ssh クライアントから接続できなくなる可能性があるため確認が必要
- X Window System の input ドライバが libinput に変更
 - Debian 8 jessie では evdev を採用
- Upstart は削除
 - systemd (デフォルト)、SysVinit、OpenRC が利用可能

開発関連

- debhelper 10
 - パラレルビルドがデフォルト化
 - autoreconf をデフォルトで実行するように変更
 - パッケージビルド時は dbgsym パッケージの生成をデフォルト化
 - * 生成した dbgsym パッケージは以下の apt-line を指定して取得
 - * deb http://debug.mirrors.debian.org/debian-debug/ stretch-debug main
 - dh_installdocs コマンドの --restart-after-upgrade オプションがデフォルト化
- 実行ファイルはデフォルトで PIE を有効にしてコンパイル及びリンクしている

インストーラ

- GUI インストールがデフォルト
- UEFI のセキュアブートは未対応
- screen 対応
- multiarch のインストーラは、amd64 をデフォルトでインストール
- HTTPS ミラーからパッケージのダウンロードが可能
- 全バイナリパッケージを提供する ISO ファイルは、CD イメージを廃止
 - DVD イメージ、blu-ray イメージのみの配布
 - CD イメージは、netinst 及び xfce4 のみのデスクトップ環境を収録した CD 一枚に収まる形でのみ提供

アップグレード方法

- リリースノートを一度読むことを推奨
- apt-line が "ftp://" の場合は、"http://"へ変更
- 利用中のバージョンが古い場合は debian-8 へ順番にメジャーアップグレードする
 - メジャーバージョンの飛ばしアップグレードは非対応
- debian-8.8 以降にアップグレードし、新しい kernel で起動するため reboot する
- debian-9 へアップグレードは upgrade、dist-upgrade の 2 段階で行う
 - apt-get update
 - apt-get upgrade
 - apt-get dist-upgrade
 - reboot
- 何かおかしい動作や不具合を見つけた場合は bugreport をお願いします

2.3 Debian Updates

- 2017/01/14: Updated Debian 8.7 released
- 2017/05/06: Updated Debian 8.8 released
- 2017/07/22: Updated Debian 8.9 released

- 2017/6/17: Debian 9 「Stretch」 released

- 2017/6/18: Debian GNU/Hurd 2017 released

Debian 9 Stretch がリリースされた翌日に、sid(=unstable) の snapshot として Debian GNU/Hurd 2017 をリリースした。

- 2017/7/22: Debian 9.1 released

- 2017/4/15: Debian Project Leader Elections 2017 投票締め切り

2017 年の Debian プロジェクトリーダー (DPL) を決める選挙が行われ、Chris Lamb さんが選出された。選挙における声明は、<https://www.debian.org/vote/2017/platforms/lamby> を参照。

- 2017/4/25: Shutting down public FTP services

<ftp://ftp.debian.org>、<ftp://security.debian.org> の FTP サービスが 2017/11/1 に停止する予定。HTTP サービスは継続するため、ftp を使っているユーザは apt-line を”http://”に変更が必要。

2.4 Debconf

Debconf は Debian の開発者のカンファレンスです。<https://debconf.org/>

- Debconf17 はカナダ モントリオールで 2017/8/6 - 8/12 に開催予定。
- Debconf18 は台湾 新竹市 (Hsinchu) で 2018/7/29 - 8/4 に開催予定。

2.5 Debian とは？

フリー/オープンなユニバーサルオペレーティングシステム を作成しようとするボランティアベースのプロジェクト。

ディストリ	企業	ボランティア
RHEL	RedHat	なし
CentOS	RedHat	あり
Ubuntu	Canonical	あり
Debian	なし	あり

Linux カーネルだけではなく、FreeBSD や GNU/Hurd のカーネルを利用した OS も提供。

- Debian 社会契約 (オープンソースの定義の元)
- Debian フリーソフトウェアガイドライン
- Debian Policy
- Ubuntu や Raspbian といったディストリビューションのベースとなっている Debian Derivatives (Debian 派生ディストリビューション調査と協力体制の整備)

世界規模で開発が行われており、63ヶ国、約 1000 名の Debian 公式開発者が開発を行っている。パッケージメンテナや翻訳などの貢献者も入れるともっと多くの開発者が参加していることになる。

- 2017 年 7 月の時点で、最新版は Debian 9.0 (コードネーム Stretch)、パッケージ数は約 51000 を提供、公式にサポートする CPU アーキテクチャは 10。
- 約 2 年毎にリリース
- 次のリリース Debian 10 (コードネーム: Buster) は 2019 年にリリースすると思われる
- コードネームはトイ・ストーリーのキャラクターを採用している。

どこで使われているのか？

Linux ディストリビューションのベースや Web サーバ、組込デバイスのベース OS として利用されている

- <http://w3techs.com/technologies/details/os-linux/all/all> <http://beagleboard.org/> <https://www.>

raspberrypi.org/

具体的には

- ISS (国際宇宙ステーション) <https://ja.wikipedia.org/wiki/>
- Steam (ゲーム PC OS)
- NAS、ルータ
- etc..

まとめると

- Debian はフリー/オープンなオペレーティングシステム (OS) を作成しようとするボランティアベースのプロジェクト。
- 自分たちの考えるフリーという言葉に関する定義、開発目的、パッケージングポリシーを厳格に決めている。
- 世界中に 1000 人以上の開発者がおり、他のディストリビューションのベースとして採用されている。
- 約 2 年毎にリリースが行われ、多くのパッケージとアーキテクチャをサポートしている。次期リリースは 2019 年になる。
- 上記のような特徴から様々なところで利用されている Linux ディストリビューションである。

2.6 Debian JP Project とは？

- 日本で Debian を普及させることを目的とした任意団体。
- Debian の日本語による情報発信、ユーザとの情報交換、Debian 開発者、パッケージメンテナの育成など。

2.7 Debian 勉強会

- 2005 年 1 月開始
- Debian Developer 上川さん発起人
- 東京と関西で月に一回コンスタントに開催している Debian 開発者、ユーザによる勉強会。

2.8 Debian 勉強会:解決したい内容

- 問題
 - ML と IRC で情報交換していた
 - face-to-face であう場所がない
 - まとまったドキュメントが出てこない
- Debian 勉強会の提案
 - 定期的に集まる
 - 資料を作成する。(GPL で！)

[git://anonscm.debian.org/tokyodebian/monthly-report.git](https://anonscm.debian.org/tokyodebian/monthly-report.git)

2.9 Debian 勉強会:実際

- Debian Weekly News Quiz
- Debian 界隈やパッケージング関連の話題など専門の人に話を聞く
- Debian 9 Stretch リリースパーティ (東京, 大阪):
 - 「Debian Continuous Integration」(Charles Plessey さん)
 - 「Debian Policy ドキュメント改良の試み」(やまねひでき さん)
 - 各地のイベントで Debian 普及活動
 - OSC2016 群馬、OSC2016 沖縄、OSC2017 北海道など
 - Debian/Ubuntu ユーザミートアップ in 札幌を開催

2.10 日本語による Debian の情報

- Debian JP Project
<http://www.debian.or.jp>
- 東京エリア Debian 勉強会
<http://tokyodebian.alioth.debian.org>
- 関西エリア Debian 勉強会

<https://wiki.debian.org/KansaiDebianMeeting>

- Twitter
@debian_jp
- G+ コミュニティ
<https://plus.google.com/u/0/communities/106942835439686570073>

3 debhelper 10 の新機能を確認してみよう

杉本 典充



3.1 はじめに

2017 年 1 月に Debian 9 (コードネーム「stretch」) の soft freeze が行われました。そのリリースの中に新しいバージョンである debhelper 10 が含まれています。

debian パッケージをビルドするツールとして普及している debhelper のバージョン 10 の新機能について調べてみました。

3.2 debhelper とは

debhelper とは、debian パッケージを作成するために役に立つツールコマンド群です。debhelper のコマンド群の多くは dh で始まるコマンド名になっています。debhelper のツール群のほとんどは perl で書かれています。

debhelper には大まかに以下の機能があります。

- パッケージをビルドできるように debian ディレクトリ配下のファイルを生成、編集するコマンド
 - 例: dh_make^{*1}、dch^{*2}
- パッケージのビルド処理に利用するコマンド
 - 例: dh^{*3}、dh_clean^{*4}、dh_install^{*5}など
 - 上記コマンドをオーバーライドしてカスタマイズすることができます (例: override_dh_clean)

3.3 debhelper の使い方例

GNU の web サイトに hello というお手本プログラムがありますので、これを例にパッケージを作成してみます。debhelper を使ったパッケージのビルドのマニュアルは以下にありますので参考にしてください。

なお、debian/rules ファイルの中身は debian パッケージ「hello」のファイルを引用しています。

<https://www.debian.org/doc/manuals/maint-guide/first.ja.html>

^{*1} パッケージをビルドする make ファイルに相当する debian/rules ファイルのひな形を作成するコマンド

^{*2} debian/changelog を編集するため EDITOR を起動するコマンド

^{*3} debian/rules のターゲットにこの 1 行だけあれば良い場合もある強力なコマンドです

^{*4} debian パッケージをビルドするために生成する build ディレクトリや不要なファイルを削除するコマンド

^{*5} ビルド後のパッケージのインストール処理を実行するコマンド

```

$ vi ~/.bashrc
export DEBEMAIL='dictoss@live.jp'
export DEBFULLNAME='Norimitsu Sugimoto'

$ sudo apt-get install dh-make build-essential

$ sudo apt-get install autotools-dev
$ wget https://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
$ tar xf hello-2.10.tar.gz
$ cd hello-2.10
$ dh_make -f ../hello-2.10.tar.gz
Type of package: (single, indep, library, python)
[s/i/l/p]? s
Email-Address      : dictoss@live.jp
License            : blank
Package Name       : hello
Maintainer Name    : Norimitsu Sugimoto
Version            : 2.10
Package Type       : single
Date               : Thu, 19 Jan 2017 23:13:10 +0900
Are the details correct? [Y/n/q] Y
Done. Please edit the files in the debian/ subdirectory now.

$ ls .. | grep hello
hello-2.10
hello-2.10.tar.gz
hello_2.10.orig.tar.gz

$ vi debian/changelog
$ vi debian/control

$ vi debian/rules
#!/usr/bin/make -f
%:
    dh $@

override_dh_auto_build:
    touch man/hello.1
    dh_auto_build

override_dh_auto_clean:
    [ ! -f Makefile ] || $(MAKE) distclean

override_dh_installdocs:
    dh_installdocs NEWS

$ dpkg-buildpackage -uc -us

$ ls .. | grep hello
hello-2.10
hello-2.10.tar.gz
hello-dbgSYM_2.10-1_amd64.deb
hello_2.10-1.debian.tar.xz
hello_2.10-1.dsc
hello_2.10-1_amd64.buildinfo
hello_2.10-1_amd64.changes
hello_2.10-1_amd64.deb
hello_2.10.orig.tar.gz

```

3.4 debhelper 10 の変更点

3.4.1 debian/compat

debhelper を使ってビルドするパッケージの場合、debian/compat ファイルを用意することになっています。debian/compat ファイルには debhelper の互換性レベルを記述し、debhelper 10 では”10”と記述します。

```

$ cat debian/compat
10

```

3.4.2 autoreconf が自動的に実行されるようになった

debian/compat=10 の場合、自動的に autoreconf が実行されるようになりました。

以下は unstable 環境の debhelper-10.2.3 において、hello パッケージをビルドしたときに出力する hello_2.10-1_amd64.build を debian/compat=9 の場合と debian/compat=10 の場合の差分を抽出した一部です。dh_autoreconf を呼び出していることを確認できます。そのため、debian/control の Build-Depends に autoreconf の実行に不足があると autoreconf でエラーが発生する場合がありますのでビルドログを確認しましょう。

```

(省略)
+ dh_autoreconf_clean
  dh_clean
  dpkg-source -b hello-2.10
dpkg-source: info: using source format '3.0 (quilt)'
@@ -64,6 +23,82 @@
dh build
  dh_testdir
  dh_update_autotools_config
+ dh_autoreconf
(省略)

```

3.4.3 Build-Depends に dh-systemd が不要になった

debhelper 9 までは systemd の処理を行うパッケージの場合は、Build-Depends に dh-systemd^{*6}を含める必要がありました。debhelper 10 では dh-systemd が取り込まれたため、debian/compat=10 の場合は Build-Depends に dh-systemd を含める必要がなくなりました。代わりに debhelper のバージョンを指定する必要があります。

```

$ cat debian/compat
10
$ cat debian/control
(省略)
Build-Depends: debhelper (>= 9.20160709)
(省略)

```

3.4.4 parallel オプションが自動的に付与されるようになった

debian/compat=10 の場合、"-parallel" オプションをデフォルトで付与するようになりビルド処理にマルチコアを有効活用するようになりました。debian/compat が 9 以下の場合には"-parallel" オプションは明示的に指定する必要がありました。

"-parallel" オプションが有効の場合、パッケージのビルド処理を行う dpkg-buildpackage コマンドの"-j" オプション (ジョブ数を指定するパラメータ) を指定するようになります。指定するジョブ数はデフォルトで論理 CPU コア数となり、"-max-parallel" オプションで明示的に指定することもできます。

また、複数ジョブを利用したビルドを意図的に行わないようにするには"-no-parallel" というオプションがあります。

```

#!/usr/bin/make -f
%:
    dh $@ --no-parallel

```

なお、hello パッケージの debian/rules は以下のようになっています。

```

#!/usr/bin/make -f
%:
    dh $@

override_dh_auto_clean:
    [ ! -f Makefile ] || $(MAKE) distclean

override_dh_installdocs:
    dh_installdocs NEWS

```

上記の debian/rules で debian/compat=9 の場合と debian/compat=10 の場合のビルドログからジョブ数に何が指定されているか確認します。

ビルドに利用したマシンは HP EliteBook 820 G3 で CPU は「Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz」となっています。2 コア HyperThreading ありの CPU であり、論理 4 コアです。以下のビルドログでは、ジョブ数が 4 になっていることを確認できます。

```

$ diff -u hello_2.10-1_amd64.build.compat9 hello_2.10-1_amd64.build.compat10 | grep 'make -j'
-make -j1
+make -j4
-make -j1 check VERBOSE=1
+make -j4 check VERBOSE=1
-make -j1 install DESTDIR=/home/norimitu/mkdeb/hello-2.10/debian/hello AM_UPDATE_INFO_DIR=no
+make -j4 install DESTDIR=/home/norimitu/mkdeb/hello-2.10/debian/hello AM_UPDATE_INFO_DIR=no

```

3.4.5 dh_installdeb の変更

debian/compat=10 では、dh_installdeb の動作の一部が変更になります。

^{*6} dh_systemd_enable、dh_systemd_start、systemd2init コマンドをインストールするパッケージ

package.maintscript の引数がシェル展開されなくなる

dh_installdeb の package.maintscript の処理時に記述するスクリプトは、シェル変数をエスケープするようになりました。以下は man dh_installdeb(1) の記述の抜粋です。

```
package.maintscript
Lines in this file correspond to dpkg-maintscript-helper(1) commands and
parameters. However, the "maint-script-parameters" should not be
included as debhelper will add those automatically.

Example:

# Correct
rm_conffile /etc/obsolete.conf 0.2~ foo
# INCORRECT
rm_conffile /etc/obsolete.conf 0.2~ foo -- "$@"
```

上記の"\$@"の部分がエスケープされるようになりシェル展開されなくなります。そのため、引数にシェル変数やシェルのメタ文字を指定していた場合は意図した文字列に置換されなくなりますので修正が必要になります。

package.shlibs の処理が dh_makeshlibs で実行するように変更

debian/compat が 9 以下の場合、dh_installdeb の処理で package.shlibs を通常/var/lib/dpkg/info ディレクトリ配下にインストールします。

debian/compat が 10 以降では、dh_makeshlibs の処理で package.shlibs を処理するように変更されました。

3.5 まとめ

debhelper 10 の新機能についてまとめてみました。stretch には debhelper 10 が含まれるためこれからパッケージを作成する場合や修正する場合に debhelper 10 へ対応させるとよいと思います。

3.6 参考文献

- debhelper 10 is now available <https://nthykier.wordpress.com/2016/09/11/debhelper-10-is-now-available/>
- man debhelper(7)
- man dh(1)
- man dh_installdeb(1)
- Debian 新メンテナーガイド <https://www.debian.org/doc/manuals/maint-guide/>
- 「dh-systemd は debhelper 9.20160709 で統合された」 henrich, 2016-08-25 <http://qiita.com/henrich/items/e1651e3284c6b3d0d39e>

4 qmake な Qt アプリの deb を作ろう として試行錯誤した話」

小林 克希



4.1 はじめに

昨年の 6 月に開催された関西 Debian 勉強会^{*7}に参加して、生産性向上のための手法の一つであるポモドーロ・テクニック^{*8}用の、Qt で書かれた tomighty ^{*9} という名前のツールのパッケージングに挑戦してみました。

私の作業前の状態は、かなーり昔に Autotools で書かれたソフトの deb パッケージをつくったことはある、といった感じでした。なので、実際のところ、このツールをひとまず deb にするところまではそこまで苦労しなかったのですが、

- lintian の警告がいっぱい出る!!
- qmake のプロジェクトでは debian/rules はどう書くのがお作法?
- GitHub+TravisCI とかどう使うの.....?

などの課題が発生したので、ここでは、それらを解決するためにやってみたり調べた内容を書かせていただこうと思います。なお、全ては解決に至っておりませんので、ヒントとか色々頂ければうれしいなあ、と思っております。どうぞよろしく願います。

4.2 tomighty

4.2.1 どんなツール?

まずは今回パッケージングしたツール、その名も“tomighty”の紹介をします。が、前述した通り、こちらはポモドーロ・テクニック用のツールですので、まずはポモドーロ・テクニックについて説明しておきます。このポモドーロ・テクニックは、生産性向上のための手法で、ざっくりと以下の流れで作業を行なう事が推奨されています。

1. 完了させたいタスクを選択して紙に書き出す
2. ポモドーロ (タイマー) を 25 分にセットする
3. タイマが鳴るまで集中する (やる事変えるのは OK)
4. タイマが鳴ったら紙にチェックを入れる (ここまでの 25 分が 1 ポモドーロという単位)
5. short-break(5 分程度?) を取る、もしくは 4 回ポモドーロを完了していたら long-break(20 ~ 30 分?) を取る

この、“1 ポモドーロ=25 分”というのがミソでして、集中してやるのに (個人的な主観で) 長すぎず、短かすぎず、ちょうど良いくらいの時間で、後々そのタスクにかけて時間の単位としても使用できる、という感じの手法になっております。とか書いてますが、私も触りしか知らないのので、ポロが出ないこの辺りで止めておきますので、興味が沸きましたら公式のドキュメント等を読んでいただければと思います。

で、tomighty はなにかというと、ツールバーに表示されるタイプのポモドーロ専用タイマアプリで、

- 1 ポモドーロの 25 分
- 休憩時間の 5 分 (short) もしくは 15 分 (long)

を計測できます。ちなみに、ポモドーロ (pomodoro) というのはイタリア語でトマトの意味で、手法の提唱者の Francesco Cirillo 氏が使っていたキッチンタイマーがトマトの形だったからこの名前になったようです。ちなみに、tomighty のアイコンもトマトですし、名前も tomato から来てるものと思われます。

^{*7} <https://wiki.debian.org/KansaiDebianMeeting/20160626>

^{*8} <http://cirillocompany.de/pages/pomodoro-technique/>

^{*9} <http://tomighty.org/>

4.2.2 まずはビルドしてみる

では、ひとまず普通にビルドしてみたいと思います。tomighty のコードは GitHub にて公開されています*¹⁰ ので、適当なディレクトリで以下のようにコマンドを叩いて取得します。

```
% git clone -b develop https://github.com/tomighty/tomighty
Cloning into 'tomighty'...
remote: Counting objects: 4621, done.
remote: Total 4621 (delta 0), reused 0 (delta 0), pack-reused 4621
Receiving objects: 100% (4621/4621), 1.83 MiB | 378.00 KiB/s, done.
Resolving deltas: 100% (2023/2023), done.
```

ここでは、clone しつつ develop ブランチを checkout しています。develop ブランチを開発用として運用しているプロジェクトは結構あるかと思いますが、このプロジェクトは master が Java で書かれた全く別物で、README にも「master の Java コードは開発を中止していて、現在 develop ブランチにて Qt5 で書き直しているよ*¹¹」となっております.....。

コードの取得が完了したら、まずは以下の通りにして新規ディレクトリ中に Makefile を作成します。qmake は、基本的にビルド用に作ったディレクトリ上に .pro ファイルの設定から Makefile を生成してビルドを行なう形になるようです。

```
% cd tomighty
% mkdir build
% cd build
% qmake -qt=qt5 ../src/tomighty.pro
Info: creating stash file /tmp/tomighty/build/.qmake.stash
```

qmake と同じように、ビルド用のディレクトリを掘って Makefile を生成するツールには cmake もあるようで、こちらも Qt のアプリ用によく使われているようです。qmake の利点—と言いますか、cmake との使い分けは、少し調べた感じだと Qt Creator*¹² と呼ばれる Qt の IDE を使うかどうか、というのが一つ大きな理由としてありそうでした。詳しくは調べきれませんでしたので、もっと他に何かあれば教えていただきたい。

ところで、qmake にさりと -qt=qt5 というオプションを使っていますが、これは複数バージョンの Qt を共存させるために qtchooser という仕組みが upstream の時点から入っており、各コマンドを叩くときに -qt オプション、もしくは QT_SELECT という環境変数でバージョンを指定することができるようです。各コマンドの名前のシンボリックリンクを作成するという、組み込み系でよく使われる busybox みたいな*¹³ 仕組みのようです。以下のコマンドを叩くとおもしろいかと。

```
% ls -l /usr/bin | grep qtchooser
...
lrwxrwxrwx 1 root root 9 11月 11 00:29 qmake -> qtchooser*
...
-rwxr-xr-x 1 root root 43728 11月 11 00:29 qtchooser*
...
```

さて、まずはと make をしてみたのですが.....、ベロベロとログを吐いた挙句、以下のようにエラーが出ました.....。

```
tomighty/build/test-runner/./core-tests//libtomighty-core-tests.so:
'tmty::InMemoryPreferences::InMemoryPreferences(QObject*)' に対する定義されていない参照です
tomighty/build/test-runner/./core-tests//libtomighty-core-tests.so:
'tmty::FakeTimer::interval() const' に対する定義されていない参照です
```

「なんですかあ？ こっちの環境の問題かなあ？」と思いつつ、以下のパッチを作成してビルドが通る事は確認しました。

```
diff --git src/test-runner/test-runner.pro src/test-runner/test-runner.pro
index 44ac2b4..82bc4c8 100644
--- src/test-runner/test-runner.pro
+++ src/test-runner/test-runner.pro
@@ -29,2 +29,3 @@ win32:CONFIG(release, debug|release): LIBS += \
- L$$OUT_PWD/./core-tests/release/ -ltomighty-core-tests \
+ L$$OUT_PWD/./core-mock/release/ -ltomighty-core-mock \
- L$$OUT_PWD/./ui-tests/release/ -ltomighty-ui-tests
@@ -35,2 +36,3 @@ else:win32:CONFIG(debug, debug|release): LIBS += \
- L$$OUT_PWD/./core-tests/debug/ -ltomighty-core-tests \
+ L$$OUT_PWD/./core-mock/debug/ -ltomighty-core-mock \
- L$$OUT_PWD/./ui-tests/debug/ -ltomighty-ui-tests
@@ -41,2 +43,3 @@ else:unix: LIBS += \
- L$$OUT_PWD/./core-tests/ -ltomighty-core-tests \
+ L$$OUT_PWD/./core-mock/ -ltomighty-core-mock \
- L$$OUT_PWD/./ui-tests/ -ltomighty-ui-tests
```

なお、コケたのはテスト用のビルドで、代わりに

*¹⁰ <https://github.com/tomighty/tomighty>

*¹¹ 実は、develop ブランチの開発も止まっているという哀しい現実が待ち構えています。

*¹² <https://www.qt.io/ide/>

*¹³ 私が組み込み畑の人間なのでこういう表現になってますが、本当はもっと良い表現があるかもしれません.....。

```
% make sub-app-all sub-core-all sub-ui-all
```

としてやれば、一応アプリ自体はビルドできます。windows の mingw とかだとエラーが出ないようです。疑問を持ちつつも自分でパッチを書いてりして deb 化の作業をすすめていたのですが、よくよく見ると、github のトップページのプロジェクトの説明欄に (私にとって) 残念な記載が.....!!

Old Tomighty repo. For the new repos, please refer to: tomighty-osx and tomighty-windows

どうやら、今回 Qt アプリとしてさわってはみたものの、開発は Objective-C で書かれた Mac 用と C# で書かれた Windows 用しか行なわれていないようです。「“Fork me on GitHub” リボンでこのプロジェクトにリンクしといて、それはないじゃーーん」と思わなくもないのですが.....。ひとまず、deb 化の練習には問題がない (?) ので、次に進めていこうかと思います.....。

4.3 deb の作成

ようやくここから debian パッケージ作成の開始です。まずは dh_make を実行するところからなのですが、upstream が GitHub でタグもまだ付けていてくれない場合はどうするのか.....結局のところわかりませんでした。git のハッシュをバージョン名に入れているパッケージは数あれど、パッケージによって結構表記が違うようで.....。どなたか教えてください.....。

結局、今回は lua-torch-doc とか lua-torch-sundown パッケージ (by Debian Science Maintainers) で使われている "0 (日付)-g(git hash)" が一番しっくり来たので、以下のようにしました。0f673359 の部分が deb 化する前の upstream の最終コミットの git hash といった感じです。

```
% dh_make --createorig -p 'tomighty_0.0.0+git0f673359' -s -c apache
```

さて、ひとまずわりくりビルドだけ通したくて、debian/rules を以下のように記述してみました。色々なターゲットを大胆に上書いています。これで、一応ビルドは通ります.....。

```
#!/usr/bin/make -f

override_dh_auto_configure:
    mkdir build
    cd build; qmake -qt=qt5 ../src/tomighty.pro

override_dh_auto_build:
    cd build; make

override_dh_auto_clean:
    rm -rf build

override_dh_auto_install:
    cd build; make install INSTALL_ROOT=$(PWD)/debian/tomighty

%:
    dh $@
```

が、もちろん決め打ち部分が多すぎてかなりイケてないのは私でもわかります。せっかく dh コマンドを使っているのに override した上に、普通にコマンドを叩いてしまっていますし.....。ということで、色々調べていくうちに、Common Debian Build System(CDBS)^{*14} に行きあたりました。こちらは、debian/rules の記述を色々ともジュール化して再利用を狙ったものということで、ばっちり qmake 用の make も存在していました。で、CDBS を使って書き直したものが以下になります。

```
QT_SELECT = qt5
DEB_QMAKE_ARGS = ../src/tomighty.pro

DEB_SRCDIR = $(CURDIR)/src
DEB_BUILDDIR = $(CURDIR)/build

cleanbulldir::
    -rm -rf $(DEB_DESTDIR)

clean::
    -rm $(DEB_BUILDDIR)/files
    -rm $(CURDIR)/debian/tomighty.substvars

include /usr/share/cdb/1/rules/debhelper.mk
include /usr/share/cdb/1/class/qmake.mk
```

clean まわりがまだ怪しいのですが、だいたすすっきり書けたのではないのでしょうか。.....と、そこまで書いてみから「もしかして.....」と調べたら、実は dh もしっかり qmake に対応していました.....。ドキュメントは見当たらなかったのですが、/usr/share/perl5/Debian/Debhelper/Buildsystem/qmake.pm を見て以下のように記載しました。

*14 <https://build-common.alioth.debian.org/cdb/1/doc.html>


```
#!/usr/bin/make -f
%:
    dh $@ --buildsystem=qmake --builddirectory=build --sourcedirectory=src

override_dh_auto_configure:
    (export QT_SELECT=qt5; dh_auto_configure -- $(CURDIR)/src/tomighty.pro)
```

override しちゃってるのが気になりますが、CDBS よりもさっくりと書いてしまいました……。もし、この override をもっと綺麗に消せる方法がありましたら、教えていただければと思います (他力本願)。ソースを眺めていると、ソースディレクトリから .pro ファイルを拾ってくれそうなんです、 --sourcedirectory オプション付けてもうまく拾ってくれませんでした……。

あと、dh_auto_configure に “export QT_SELECT=qt5” が並んでいる件ですが、これをやっておかないと pbuilder 環境で qtchooser が彷徨ってしまって以下のようなエラーになってしまいました……。

```
qmake: could not find a Qt installation of ''
```

qt5-default パッケージを入れると直るという情報があったり、でも Qt のメンテナが qtX-default にビルド依存してくれるという情報もあったり^{*15} で、ちょっと正解はわかりませんでした……。なお、Qt4 に関しては、 --buildsystem オプションを qmake から qmake_qt4 にしてあげればよさそうです……。

(後日追記) 上記の stackoverflow で Qt のメンテナのリンク先のブログ^{*16} に

- Exporting QT_SELECT with 4, qt4, 5 or qt5 as a value in debian/rules.
- Call the tool using the '-qtX' parameter, where x can be replaced with any of the options above.

とあり、QT_SELECT を export するのが正解のようです。ただ、今回それを override ルールに無理矢理記載しましたが、以下のように Makefile 的に export する方が筋が良さそうです。ありがとうございました。 > 関西 Debian の皆様

```
#!/usr/bin/make -f
export QT_SELECT=qt5
%:
    dh $@ --buildsystem=qmake --builddirectory=build --sourcedirectory=src

override_dh_auto_configure:
    dh_auto_configure -- $(CURDIR)/src/tomighty.pro
```

4.4 gbp を使う

upstream が git で管理されている場合、gbp (git-buildpackage) ^{*17} を使うのがよいらしいので、ひとまず今回のパッケージングでも gbp を使ってみることにしました。とはいいつつも、ごめんなさい、今回 gbp については全然使いこなせておりません。

とりあえず、gbp の設定をしたいと思います。今回、前述した通り upstream の branch は develop となっています。deb を作成するにあたっては、通常は debian という名前で branch を切って、その中で dh_make で debian ディレクトリを作ってあげるのが良いようです。今回は、先に develop branch で dh_make しちゃってましたが、特にコミットはしていないので git checkout -b debian でよろしかったかと。

続けて、gbp の設定ファイルを作ります。ユーザ毎の設定は ~/.gbp.conf に、deb 用のリポジトリ毎の設定なら、debian/gbp.conf につくるのがよさそうです。今回は、以下のように upstream と debian の branch を設定しました。

```
[DEFAULT]
upstream-branch=develop
debian-branch=debian
```

この状態で、未コミットの物がある場合は --git-ignore-new オプションを付けつつ buildpackage を実行すると、orig.tar.gz を含めて作成してくれました。

4.5 travis.debian.net を使う

せっかく deb を作れるようになったので、できれば CI ツールを使いたいと思い、GitHub なら Travis CI ^{*18} が有名なのかなと調べてみたところ、travis.debian.net ^{*19} というのがあるようです。こちらは、ビルド依存しているパッケージを含めた Debian

^{*15} <http://stackoverflow.com/questions/16607003/qmake-could-not-find-a-qt-installation-of>

^{*16} <http://perezmeier.blogspot.jp/2013/08/qt-in-debian-using-qt4-andor-qt5-in.html>

^{*17} <https://honk.sigxcpu.org/piki/projects/git-buildpackage/>

^{*18} <https://travis-ci.org/>

^{*19} <http://travis.debian.net/>

の Docker イメージを作成してビルドのテストを行なってくれるようです。なお、gbp でビルドできる事が前提のようです.....。

設定は比較的簡単で、

1. Travis CI でテストしたいリポジトリのビルドを有効にし、“Build only if .travis.yml is present” である事を確認しておく。
2. リポジトリのルートディレクトリにビルド設定ファイル .travis.yml を置く。
3. debian/source/options に extend-diff-ignore = “^\.travis\.yml\$” と書いておく

となっております。設定ファイルの.travis.yml の中身が問題なわけですが、私の場合はひとまず以下になりました。

```
sudo: required
language: c++

env:
  global:
    - TRAVIS_DEBIAN_GIT_BUILDPACKAGE="gbp buildpackage --git-upstream-tag=6afa215"
    - TRAVIS_DEBIAN_DISTRIBUTION="sid"
    - TRAVIS_DEBIAN_EXTRA_REPOSITORY_GPG_URL=""
    - TRAVIS_DEBIAN_EXTRA_REPOSITORY=""

services:
  - docker

script:
  - wget -O- http://travis.debian.net/script.sh | sh -

branches:
  except:
    - /^debian\/\d/
```

upstream の指定がうまくできず、ひとまず --git-upstream-tag オプションにて無理矢理コミットのハッシュ値つっこみます.....。いずれ、良い感じに直したい.....。

yml のファイルを見て、「“wget | sh -” て!!」と思われた方もいらっしゃると思いますが、それについては travis.debian.net の FAQ でも一応触れられていて、「自分のマシンなら、こんなコマンドは実行すべきではないけど、使い捨てのコンテナの中で実行されて、生成された.deb も使われないから良いんだ」的な感じでまとめられています。ひとまず、ここまで設定すると、GitHub にコミットしたタイミングで TravisCI でビルドが走るようになりました。

4.6 まとめ

ということで、若干中途半端ではありますが、ひとまずはここまでの内容で当初の課題のうち、

- qmake のプロジェクト向けな debian/rules の記述
- GitHub+TravisCI でもビルドのチェック

はできるようになりました。もちろん、改善すべき点はまだまだ色々残っていきそうだし、まだ解決に至っていない問題も多く、今後は

- gbp の使い方 (特に travis.debian.net との連携部分)
- lintian の警告の修正

あたりの確認をしていかなばと思っています。ちなみに、2 つ目は、tomighty の upstream がポドローロ・テクニック用の共有ライブラリを作るので、その辺りがまだ調査できておらず色々怒られてます.....。

なにせよ、今回、随分とひさしぶりに deb を作ってみたのですが、以前と比べて、

- dh コマンドすごい.....。
- tar.gz で配布されてるソフトで gbp はかなり便利そう。
- (今回は使わなかったですが)quilt でパッチの管理が楽になってる。

と感じました。upstream の開発が止まっているという根本的な問題があるので、今回の tomighty の deb はどうするのかというのはちょっと悩み中なのですが、今後、上記の未解決案件に加えて ITP とかの手順も調べていけたらなあと思っています。

ここまでお付き合い頂き、ありがとうございました。

5 Debian パッケージのカスタマイズについて

yy-y-ja-jp



5.1 はじめに

Debian パッケージをカスタマイズしたいけどやり方が (特にどこに書いてあるのか) 分からないという話があったので、まとめてみました。

まずパッケージングの前提を見ていきます。

次に、カスタマイズしたい場所は大きく 2 つあるでしょう。

- Debian パッケージングの部分
- 上流のコード自体

また、これらカスタマイズした内容をその後メンテナンスしていく必要があります。それぞれ見ていきましょう。

5.2 前提

パッケージングについての概要は “パッケージングチュートリアル” にあるので `apt install packaging-tutorial` を実行してから `/usr/share/doc/packaging-tutorial/` の中を見てみましょう^{*20}。

カスタマイズするにはパッケージのソースコードを手元に用意する必要があります。 `apt-get source` や `dget(1)` (`devscripts` パッケージにあります) でダウンロードできます。メンテナがパッケージング作業用の `git` レポジトリなどを公開している場合 (`Vcs-*` フィールド) は、`debcheckout(1)` (`devscripts` パッケージ) も使うとそのレポジトリを `clone` したりしてくれるようです。

そして、Debian パッケージは “Debian ポリシー” (`debian-policy` パッケージ) に従って作られています。Debian ポリシーは、Debian がうまく動くように定められたルールです。カスタマイズしたパッケージもポリシーに従ったほうがよいでしょう。本当はポリシー全文を読むとよいでしょう、と言いたいところですが長文です。また、Debian ポリシー以外にも従っておいたほうがよい慣習が “Debian 開発者リファレンス” (`developers-reference-ja` パッケージ)^{*21} などいくつか書かれています。

`debhelper` などのパッケージングヘルパーはポリシーにできるだけ自動的に従ってパッケージが作られるように仕組みであります。なので普通にパッケージングヘルパーを使ってパッケージング作業をしていればある程度はポリシーに勝手に従っているはず

です。

また、`lintian(1)` というツール (`lintian` パッケージ) が Debian ポリシーに違反していないか、慣習に沿っているかチェックしてくれます。`lintian` はインストール済みならパッケージングコマンド `debuild` の最後で自動的に呼ばれます。実際のところいろんな慣習がいろんな場所に書いてあってわからないことも多いので、まずは作ったパッケージを `debuild` してみて、`lintian` の結果を見て、エラーや警告が出たら詳細説明を読んで修正しましょう。

特に、慣習の 1 つにはバージョン番号があります。開発者リファレンスによると、パッケージメンテナ以外がパッケージを変更して Debian 本体にアップロードするとき (Non-Maintainer Upload, NMU) には、変更内容を `debian/changelog` に書き、特別なバージョン番号にしなければなりません。常にメンテナのアップロードするバージョンが優先されるように、次にメンテナがアップロードするであろうバージョン番号よりも低くしているからです。本当に NMU するわけではないにせよ重要なので、これについては以降も見ていきます。

^{*20} <https://www.debian.org/doc/manuals/packaging-tutorial/packaging-tutorial.ja.pdf> にもあります。

^{*21} <https://www.debian.org/doc/manuals/developers-reference/index.ja.html> にもあります。

5.3 Debian パッケージングの部分

これについては

- コンパイルオプションなどを変更したい
- 依存関係やパッケージ構成などを変更したい
- 不安定版 (unstable) にあるバージョンを安定版 (stable) などにバックポートしたい

などがあるでしょう。

5.3.1 コンパイルオプション

パッケージングでは `debian/rules` ファイルからコンパイルが実行されます。コンパイルオプションを変更するときは、これを変更するだけのことが多いです (正確には、一通り変更して、その変更内容を `dch(1)` コマンドで `debian/changelog` ファイルに新しいバージョン番号で記入したら完成です。以降も同様です)。

ここでは例として `hello` というパッケージを見てみます。

```
$ apt-get source hello
パッケージリストを読み込んでいます... 完了
733 kB のソースアーカイブを取得する必要があります。
取得:1 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (dsc) [1,323 B]
取得:2 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (tar) [726 kB]
取得:3 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (diff) [6,072 B]
(snip)
$ cd hello-2.10/
```

```
$ editor debian/rules
```

```
#!/usr/bin/make -f
%:
    dh $@

override_dh_auto_clean:
    [ ! -f Makefile ] || $(MAKE) distclean

override_dh_installdocs:
    dh_installdocs NEWS
```

この `hello` にはあまり面白そうなオプションが見当たらないのですが、とりあえず `./configure` に `--disable-nls` を付けてみます。さて、どこに書けばいいかわからないのでとりあえず `debuild` してみます。

```
$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1
dpkg-buildpackage: info: source distribution unstable
(snip)
debian/rules build
dh build
dh_testdir
dh_update_autotools_config
dh_auto_configure
./configure --build=x86_64-linux-gnu --prefix=/usr --includedir=\${prefix}/include \
--mandir=\${prefix}/share/man --infodir=\${prefix}/share/info --sysconfdir=/etc \
--localstatedir=/var --disable-silent-rules --libdir=\${prefix}/lib/x86_64-linux-gnu \
--libexecdir=\${prefix}/lib/x86_64-linux-gnu --disable-maintainer-mode --disable-dependency-tracking
configure: WARNING: unrecognized options: --disable-maintainer-mode
(snip)
```

どうやら `dh_auto_configure` の `man` ページを読むと、これにオプションを渡せばいいようなのでそうしてみます。

```
#!/usr/bin/make -f
%:
    dh $@

override_dh_auto_clean:
    [ ! -f Makefile ] || $(MAKE) distclean

override_dh_installdocs:
    dh_installdocs NEWS

override_dh_auto_configure:
    dh_auto_configure -- --disable-nls
```

debian/changelog を書きます。dch を実行すると*²²、メンテナではないため自動的に--nmu モードになります。ディストリビューションが UNRELEASED に仮で書かれます。

```
$ dch
$ head -n 10 debian/changelog
hello (2.10-1.1) UNRELEASED; urgency=medium

 * Non-maintainer upload.
 * debian/rules: Use --disable-nls.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 08:58:38 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.
```

とりあえず debuild してみます。最後に lintian が実行されてるので見ましょう。

```
$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1.1
dpkg-buildpackage: info: source distribution unstable
(snip)
 dh_md5sums
 dh_builddeb
dpkg-deb: building package 'hello-dbg' in './hello-dbg_2.10-1.1_amd64.deb'.
dpkg-deb: building package 'hello' in './hello_2.10-1.1_amd64.deb'.
dpkg-genbuildinfo
dpkg-genchanges >./hello_2.10-1.1_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.
```

lintian の各行の詳細説明は lintian-info(1) で見られます*²³。

```
$ lintian-info --tags ancient-standards-version
W: ancient-standards-version
N:
N: The source package refers to a Standards-Version that has been
N: obsolete for more than two years. Please update your package to latest
N: Policy and set this control field appropriately.
N:
N: If the package is already compliant with the current standards, you
N: don't have to re-upload the package just to adjust the
N: Standards-Version control field. However, please remember to update
N: this field next time you upload the package.
N:
N: See /usr/share/doc/debian-policy/upgrading-checklist.txt.gz in the
N: debian-policy package for a summary of changes in newer versions of
N: Policy.
N:
N: Refer to https://www.debian.org/doc/debian-policy/upgrading-checklist
N: for details.
N:
N: Severity: normal, Certainty: certain
N:
N: Check: standards-version, Type: source
N:
```

なお、この例ではカスタマイズする前から ancient-standards-version が出ているようなので、修正せずそのままにしておきます。メンテナが次にリリースしたときに直すでしょうし、それに対して自分のカスタマイズをしたいときに、自分のカスタマイズ内容だけを変更すればよいからです。

カスタマイズが完了したらディストリビューションを確定させます。以下を実行してエディタで保存します。

*²² 実行前に DEBFULLNAME, DEBEMAIL 環境変数を設定してください。詳しくは“新メンテナガイド”(後述)にあります。

*²³ <https://lintian.debian.org/tags/ancient-standards-version.html> でも見られますが、問題のある全パッケージが載っているので重いです。

```

$ dch -r
$ head -n 10 debian/changelog
hello (2.10-1.1) unstable; urgency=medium

 * Non-maintainer upload.
 * debian/rules: Use --disable-nls.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 09:00:30 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.

```

debuild すれば完成です。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1.1
dpkg-buildpackage: info: source distribution unstable
(snip)
  dh_md5sums
  dh_builddeb
dpkg-deb: building package 'hello-dbgSYM' in '../hello-dbgSYM_2.10-1.1_amd64.deb'.
dpkg-deb: building package 'hello' in '../hello_2.10-1.1_amd64.deb'.
dpkg-genbuildinfo
dpkg-genchanges >../hello_2.10-1.1_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

さて、ここまで (dch と引数なしで起動することで) NMU するパッケージング作業をしてきました。Debian 本体にアップロードするのではなく個人的に使ったりするためのカスタマイズであれば、NMU ではなく “ローカルアップロード” にしたほうがよいでしょう。ローカルアップロードには名前を付けます。ここでは名前を ysn にしてみます。

```
$ dch --local ysn
```

```

$ head -n 10 debian/changelog
hello (2.10-1ysn1) UNRELEASED; urgency=medium

 * debian/rules: Use --disable-nls.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 08:44:10 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.
 * debian/patches: Drop 01-fix-i18n-of-default-message, no longer needed.
(snip)

```

debuild してみます。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1ysn1
dpkg-buildpackage: info: source distribution UNRELEASED
(snip)
  dh_md5sums
  dh_builddeb
dpkg-deb: building package 'hello-dbgSYM' in '../hello-dbgSYM_2.10-1ysn1_amd64.deb'.
dpkg-deb: building package 'hello' in '../hello_2.10-1ysn1_amd64.deb'.
dpkg-genbuildinfo
dpkg-genchanges >../hello_2.10-1ysn1_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
Now running lintian...
W: hello source: changelog-should-mention-nmu
W: hello source: source-nmu-has-incorrect-version-number 2.10-1ysn1
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

lintian が警告しているようなので見てみます。

```

$ lintian-info --tags changelog-should-mention-nmu source-nmu-has-incorrect-version-number
W: changelog-should-mention-nmu
N:
N:   When you NMU a package, that fact should be mentioned on the first
N:   line in the changelog entry. Use the words "NMU" or "Non-maintainer
N:   upload" (case insensitive).
N:
N:   Maybe you didn't intend this upload to be a NMU, in that case, please
N:   double-check that the most recent entry in the changelog is
N:   byte-for-byte identical to the maintainer or one of the uploaders. If
N:   this is a local package (not intended for Debian), you can suppress
N:   this warning by putting "local" in the version number or "local
N:   package" on the first line of the changelog entry.
N:
N:   Refer to Debian Developer's Reference section 5.11.3 (Using the
N:   DELAYED/ queue) for details.
N:
N:   Severity: normal, Certainty: certain
N:
N:   Check: nmu, Type: source
N:

```

```

W: source-nmu-has-incorrect-version-number
N:
N:   A source NMU should have a Debian revision of "-x.x" (or "+nmux" for a
N:   native package). This is to prevent stealing version numbers from the
N:   maintainer.
N:
N:   Maybe you didn't intend this upload to be a NMU, in that case, please
N:   double-check that the most recent entry in the changelog is
N:   byte-for-byte identical to the maintainer or one of the uploaders. If
N:   this is a local package (not intended for Debian), you can suppress
N:   this warning by putting "local" in the version number or "local
N:   package" on the first line of the changelog entry.
N:
N:   Refer to Debian Developer's Reference section 5.11.2 (NMUs and
N:   debian/changelog) for details.
N:
N:   Severity: normal, Certainty: certain
N:
N:   Check: nmu, Type: source
N:

```

Debian にアップロードする目的のパッケージではないので、debian/changelog を修正します。

```

$ dch
$ head -n 10 debian/changelog
hello (2.10-1ysn1) UNRELEASED; urgency=medium

 * Local package.
 * debian/rules: Use --disable-nls.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 08:44:10 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.

```

ここでは Local package と書くことにしましたが、dch --local local でもいいでしょう。debuild します。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1ysn1
dpkg-buildpackage: info: source distribution UNRELEASED
(snip)
dh_md5sums
dh_builddeb
dpkg-deb: building package 'hello-dbgSYM' in './hello-dbgSYM_2.10-1ysn1_amd64.deb'.
dpkg-deb: building package 'hello' in './hello_2.10-1ysn1_amd64.deb'.
dpkg-genbuildinfo
dpkg-genchanges >./hello_2.10-1ysn1_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

問題なさそうなので dch -r して debuild すれば完成です。

5.3.2 依存関係・パッケージ構成

依存関係やパッケージ構成は debian/control に書かれているので、変更するならこれを変更するだけです。

5.3.3 バックポート

バックポートするには、基本的には `dch --bpo` するだけで、他の場所を変更してはいけません。バックポート先のライブラリのバージョンが古いときは `debian/control` の依存関係を変更する必要があることがあります。

`jessie` にバックポートしてみます。まず新しいバージョンのソースパッケージを手元にダウンロードします。

```
$ apt-get source -d hello
パッケージリストを読み込んでいます... 完了
733 kB のソースアーカイブを取得する必要があります。
取得:1 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (dsc) [1,323 B]
取得:2 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (tar) [726 kB]
取得:3 http://ftp.jp.debian.org/debian sid/main hello 2.10-1 (diff) [6,072 B]
733 kB を 1 秒 で取得しました (634 kB/s)
ダウンロードオンリーモードでパッケージのダウンロードが完了しました
```

このソースパッケージをバックポート先 (`jessie`) 環境に持っていきます。

```
$ scp -p hello_2.10.orig.tar.gz hello_2.10-1.d* jessie:~/
(snip)
```

`jessie` 環境で持ってきたソースパッケージを展開して作業します。

```
jessie$ dpkg-source -x hello_2.10-1.dsc
dpkg-source: info: extracting hello in hello-2.10
dpkg-source: info: unpacking hello_2.10.orig.tar.gz
dpkg-source: info: unpacking hello_2.10-1.debian.tar.xz
jessie$ cd hello-2.10/
jessie$ dch --bpo
```

* だけの行が自動的に追加されますが、今回は特に依存関係は変えてないので削除して保存します。

```
jessie$ head -n 10 debian/changelog
hello (2.10-1~bpo8+1) jessie-backports; urgency=medium

 * Rebuild for jessie-backports.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 17:05:44 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.
 * debian/patches: Drop 01-fix-i18n-of-default-message, no longer needed.
```

`debuild` してみます。

```
jessie$ debuild -us -uc
dpkg-buildpackage -rfakeroot -D -us -uc
dpkg-buildpackage: source package hello
dpkg-buildpackage: source version 2.10-1~bpo8+1
dpkg-buildpackage: source distribution jessie-backports
(snip)
Now running lintian...
E: hello changes: backports-changes-missing
W: hello source: changelog-should-mention-nmu
W: hello source: source-nmu-has-incorrect-version-number 2.10-1~bpo8+1
Finished running lintian.
```

lintian エラーが出ています。

```
jessie$ lintian-info --tags backports-changes-missing
E: backports-changes-missing
N:
N: The changes file only has changelog entries from a single version. For
N: backports, all changes since (old)stable or the previous backport
N: should be listed (usually by adding the '-v' option to the build).
N:
N: Refer to http://backports.debian.org/Contribute/ for details.
N:
N: Severity: serious, Certainty: possible
N:
N: Check: changes-file, Type: changes
N:
```

`jessie` の最新版は `2.9-2+deb8u1` のようです。

この Debian Backports のページを見ると、`debuild -v2.9-2+deb8u1` とすればよいようです。

他の lintian 警告も修正してみます。


```

jessie$ dch
jessie$ head -n 10 debian/changelog
hello (2.10-1~bpo8+1) jessie-backports; urgency=medium

* Local package.
* Rebuild for jessie-backports.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 17:05:44 +0900

hello (2.10-1) unstable; urgency=low

* New upstream release.

```

言われたように rebuild してみます。

```

jessie$ debuild -v2.9-2+deb8u1 -us -uc
dpkg-buildpackage -rfakeroot -D -us -uc -v2.9-2+deb8u1
dpkg-buildpackage: source package hello
dpkg-buildpackage: source version 2.10-1~bpo8+1
dpkg-buildpackage: source distribution jessie-backports
(snip)
dh_builddeb
dpkg-deb: './hello_2.10-1~bpo8+1_amd64.deb' にパッケージ 'hello' を構築しています。
dpkg-genchanges -v2.9-2+deb8u1 >../hello_2.10-1~bpo8+1_amd64.changes
parsechangelog/debian: warning: 'since' option specifies non-existing version
parsechangelog/debian: warning: use newest entry that is earlier than the one specified
dpkg-genchanges: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: binary and diff upload (original source NOT included)
Now running lintian...
Finished running lintian.

```

なんか dpkg-genchanges の警告が出てますがとりあえず解消しました。

ですが、個人的には Debian で公式なバックポートがリリースされたら更新されてほしいのでちょっと不満です。lintian エラーはうまく消えないのですが、少しバージョン番号を下げて使うことにしています。

```

jessie$ dch
jessie$ head -n 10 debian/changelog
hello (2.10-1~bpo8+0.1) jessie-backports; urgency=medium

* Local package.
* Rebuild for jessie-backports.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 17:05:44 +0900

hello (2.10-1) unstable; urgency=low

* New upstream release.

```

rebuild します。

```

jessie$ debuild -v2.9-2+deb8u1 -us -uc
dpkg-buildpackage -rfakeroot -D -us -uc -v2.9-2+deb8u1
dpkg-buildpackage: source package hello
dpkg-buildpackage: source version 2.10-1~bpo8+0.1
dpkg-buildpackage: source distribution jessie-backports
(snip)
dpkg-genchanges -v2.9-2+deb8u1 >../hello_2.10-1~bpo8+0.1_amd64.changes
parsechangelog/debian: warning: 'since' option specifies non-existing version
parsechangelog/debian: warning: use newest entry that is earlier than the one specified
dpkg-genchanges: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: binary and diff upload (original source NOT included)
Now running lintian...
E: hello changes: backports-upload-has-incorrect-version-number 2.10-1~bpo8+0.1
Finished running lintian.

```

エラー消えませんが、あきらめます。

```

$ lintian-info --tags backports-upload-has-incorrect-version-number
E: backports-upload-has-incorrect-version-number
N:
N:   The version number doesn't comply with the standard backport version
N:   rules. It should end in ~bpoX+N, where X is the release version number
N:   of the target distribution.
N:
N:   Refer to http://backports.debian.org/Contribute/ for details.
N:
N: Severity: serious, Certainty: certain
N:
N: Check: changes-file, Type: changes
N:

```

lintian が示している Debian Backports のページを見ても特に書いていないので、個人的にはこのまま使うことにしています。

5.4 上流のコード自体

これについては

- コードに変更を加えたい
- 新しいバージョンなどに置き換えたい

があるでしょう。

5.4.1 コードに変更を加える

Debian パッケージのバージョンによって異なりますが、debian/source/format に 3.0 (quilt) と書いてある最近のパッケージであれば “quilt” というもので上流コードに対する変更差分を管理しています^{*24}。quilt(1) コマンドでも作業はできるのですが、変更を加えるだけなら dpkg-source --commit が使えます。使い方は 2013 年 2 月の Debian パッケージング道場資料^{*25}を見るとよいでしょう。

Debian パッケージのメンテナが git-buildpackage を使っている場合は gbp pq も使えます。2015 年 9 月の第 130 回東京エリア Debian 勉強会での Debian パッケージング道場の資料^{*26}を見るとよいでしょう。

日本語訳を書き換えてみます。

```
$ apt-get source hello
(snip)
$ cd hello-2.10/
$ editor po/ja.po
```

変更内容を debian/changelog に書きます。

```
$ dch
$ head -n 10 debian/changelog
hello (2.10-1.1) UNRELEASED; urgency=medium

 * Non-maintainer upload.
 * po/ja.po: Refine translation.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 12:02:56 +0900

hello (2.10-1) unstable; urgency=low

 * New upstream release.
```

debuild してみます。

```
$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1.1
dpkg-buildpackage: info: source distribution UNRELEASED
(snip)
dpkg-source -b hello-2.10
dpkg-source: info: using source format '3.0 (quilt)'
dpkg-source: info: building hello using existing ./hello_2.10.orig.tar.gz
dpkg-source: info: local changes detected, the modified files are:
hello-2.10/po/ja.po
dpkg-source: error: aborting due to unexpected upstream changes, see /tmp/hello_2.10-1.1.diff.sbQf7t
dpkg-source: info: you can integrate the local changes with dpkg-source --commit
dpkg-buildpackage: error: dpkg-source -b hello-2.10 gave error exit status 2
debuild: fatal error at line 1116:
dpkg-buildpackage -rfakeroot -us -uc failed
```

dpkg-source --commit するとパッチ名が聞かれます。入力するとその名前のファイルに変更差分を保存されてエディタが起動します。パッチの説明を先頭を書くことになっているので、自動生成されたテンプレートに書かれている通り “DEP-3”^{*27}に従って書いてください。

```
$ dpkg-source --commit
dpkg-source: info: local changes detected, the modified files are:
hello-2.10/po/ja.po
Enter the desired patch name: refine-ja-translation
dpkg-source: info: local changes have been recorded in a new patch: hello-2.10/debian/patches/refine-ja-translation
```

^{*24} debian/source/format がでない場合は古いバージョン 1.0 です。これは Debian パッケージングの部分と上流のコードへの変更を分離して管理していないので、5.3 節と同じ作業でできます。

^{*25} <http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201302-dojo.pdf>

^{*26} <http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume201509-presentation.pdf>

^{*27} <http://dep.debian.net/deps/dep3/>

debuild します。

```
$ debuild -us -uc
(snip)
```

さて、ここでさらにデフォルトの hello 出力を変えてみることにします。

```
$ editor src/hello.c
```

変更内容を debian/changelog に追記します。

```
$ dch
$ head -n 10 debian/changelog
hello (2.10-1.1) UNRELEASED; urgency=medium

* Non-maintainer upload.
* po/ja.po: Refine translation.
* src/hello.c: Change default message.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 12:02:56 +0900

hello (2.10-1) unstable; urgency=low
```

debuild してみます。

```
$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1.1
(snip)
dpkg-source: info: using source format '3.0 (quilt)'.
dpkg-source: info: building hello using existing ./hello_2.10.orig.tar.gz
dpkg-source: info: local changes detected, the modified files are:
hello-2.10/src/hello.c
dpkg-source: error: aborting due to unexpected upstream changes, see /tmp/hello_2.10-1.1.diff.xIhd5q
dpkg-source: info: you can integrate the local changes with dpkg-source --commit
dpkg-buildpackage: error: dpkg-source -b hello-2.10 gave error exit status 2
debuild: fatal error at line 1116:
dpkg-buildpackage -rfakeroot -us -uc failed
```

dpkg-source --commit します。何度でもできます。

```
$ dpkg-source --commit
dpkg-source: info: local changes detected, the modified files are:
hello-2.10/src/hello.c
Enter the desired patch name: change-default-message
dpkg-source: info: local changes have been recorded in a new patch: hello-2.10/debian/patches/change-default-message
```

debuild します。

```
$ debuild -us -uc
(snip)
dh_auto_test: make -j1 check VERBOSE=1 returned exit code 2
debian/rules:3: ターゲット 'build' のレシピで失敗しました
make: *** [build] エラー 2
dpkg-buildpackage: error: debian/rules build gave error exit status 2
debuild: fatal error at line 1116:
dpkg-buildpackage -rfakeroot -us -uc failed
```

テストが通らなかったのが直します。最後に保存したパッチファイルの修正が必要なので、ここからは quilt(1) の知識が必要です。詳細は 2007 年 1 月の第 24 回東京エリア Debian 勉強会資料にある “パッチ管理ツール quilt の使い方”^{*28}を見るとよいでしょう。

```
$ export QUILT_PATCHES=debian/patches
$ quilt add tests/hello-1
ファイル tests/hello-1 をパッチ debian/patches/change-default-message に追加しました
$ editor tests/hello-1
$ quilt refresh
パッチ debian/patches/change-default-message をリフレッシュしました
```

debuild して内容に満足したでしょうか。さて、ここまでバージョン番号を NMU の 2.10-1.1 にしましたが、Debian 本体でも NMU が行われて同じ 2.10-1.1 が出現してしまうかもしれません。ローカルのカスタマイズなのでもう少しバージョンを下げておくことにします。

^{*28} <http://tokyodebian.alioth.debian.org/pdf/debianmeetingresume200701.pdf>

```

$ dch
$ head -n 10 debian/changelog
hello (2.10-1.0) UNRELEASED; urgency=medium

* Non-maintainer upload.
* po/ja.po: Refine translation.
* src/hello.c: Change default message.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 12:02:56 +0900

hello (2.10-1) unstable; urgency=low

```

debuild してみます。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10-1.0
(snip)
dpkg-genchanges >../hello_2.10-1.0_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build hello-2.10
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

特に lintian 警告は出ないようなので、これでとりあえずよさそうです。dch -r して再度 debuild すれば完成です。

5.4.2 新しいバージョンなどに置き換える

uscan(1), uupdate(1) などが使えます。“新メンテナーガイド” (apt install maint-guide-ja を実行してから /usr/share/doc/maint-guide-ja/の中を見てください^{*29}) の“8. パッケージの更新”を読むとよいでしょう。

git-buildpackage を使っている場合は gbp import-orig と gbp pq も使えます。

なお、カスタマイズしているパッケージが Ruby などの言語のパッケージ (特にライブラリパッケージ) で、使っているライブラリの依存関係などが大きく更新されている場合は一からパッケージングをやり直したほうがよいこともあります。

GNU hello の最新 Git スナップショットに置き換えてみます。

```

$ git clone https://git.savannah.gnu.org/git/hello.git
(snip)
$ cd hello/
$ ./bootstrap
./bootstrap: Bootstrapping from checked-out hello sources...
(snip)
./bootstrap: done. Now you can run './configure'.
$ ./configure
(snip)
$ make check syntax-check distcheck
(snip)
=====
hello-2.10.17-4339 archives ready for distribution:
hello-2.10.17-4339.tar.gz
=====

```

```

$ uupdate ../hello-2.10.17-4339.tar.gz
update: new version number not recognized from given filename
update: Please run update with the -v option
$ uupdate ../hello-2.10.17-4339.tar.gz -v 2.10.17-4339
update: New Release will be 2.10.17-4339-1.
Symlinking to pristine source from hello_2.10.17-4339.orig.tar.gz...
update: Untarring the new sourcecode archive ../hello-2.10.17-4339.tar.gz
update: Unpacking the debian/ directory from version 2.10-1 worked fine.
update: Remember: Your current directory is the OLD sourcearchive!
update: Do a "cd ../hello-2.10.17-4339" to see the new package
$ cd ../hello-2.10.17-4339

```

debuild してみます。

^{*29} <https://www.debian.org/doc/manuals/maint-guide/index.ja.html> にもあります。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10.17-4339-1
(snip)
dpkg-genchanges >./hello_2.10.17-4339-1_amd64.changes
dpkg-genchanges: info: including full source code in upload
dpkg-source --after-build hello-2.10.17-4339
dpkg-buildpackage: info: full upload (original source is included)
Now running lintian...
W: hello source: changelog-should-mention-nmu
W: hello source: source-nmu-has-incorrect-version-number 2.10.17-4339-1
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

lintian に従って、“Debian 開発者リファレンス” も読みつつ debian/changelog を直します。あとは、まだ New upstream release でもないので直します。

```

$ dch
$ head -n 10 debian/changelog
hello (2.10.17-4339-0.1) UNRELEASED; urgency=medium

* Non-maintainer upload.
* New upstream snapshot.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 13:01:54 +0900

hello (2.10-1) unstable; urgency=low

* New upstream release.

```

debuild します。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10.17-4339-0.1
(snip)
dpkg-buildpackage: info: full upload (original source is included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

lintian は問題なさそうです。前節同様、Debian 本体でもこの Git スナップショットの NMU が出てしまうかもしれないので、さらにもう少しバージョンを下げておくことにします。

```

$ dch
$ head -n 10 debian/changelog
hello (2.10.17-4339-0.0) UNRELEASED; urgency=medium

* Non-maintainer upload.
* New upstream snapshot.

-- YOSHINO Yoshihito <yy.y.ja.jp@gmail.com> Sat, 22 Apr 2017 13:01:54 +0900

hello (2.10-1) unstable; urgency=low

* New upstream release.

```

debuild します。

```

$ debuild -us -uc
dpkg-buildpackage -rfakeroot -us -uc
dpkg-buildpackage: info: source package hello
dpkg-buildpackage: info: source version 2.10.17-4339-0.0
(snip)
dpkg-source --after-build hello-2.10.17-4339
dpkg-buildpackage: info: full upload (original source is included)
Now running lintian...
W: hello source: ancient-standards-version 3.9.6 (current is 3.9.8)
Finished running lintian.

```

lintian はとりあえず問題なさそうです。dch -r して debuild すれば完成です。

5.5 メンテナンス

カスタマイズしていたパッケージの新バージョンがそのパッケージの配布元 (Debian など) でリリースされたときは、カスタマイズ作業やり直します。

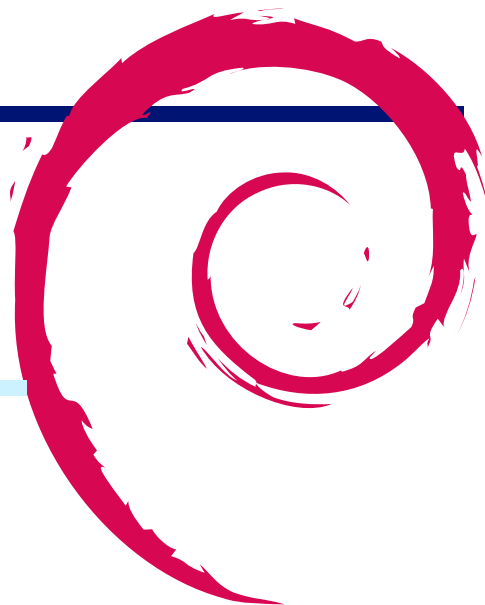
そうではないが上流コードのさらに新しいバージョンがリリースされたときは、uscan と uupdate、gbp pq などが使えます。

5.6 まとめ

Debian ポリシーなどの慣習に従うことで、Debian システムの恩恵が受けられます。そのためにポリシーなどを本当は読む必要がありますが、debhelper を素直に使っていればある程度はポリシーなどに従って作れるので、まずは作ってみて `debuild` して、`lintian` の結果を無視しないで詳細説明も読んで従いましょう。読んでもわからないところがあったら Debian JP メーリングリストや Debian 勉強会などで質問すれば答えてくれるでしょう。

5.7 参考文献

- `packaging-tutorial`
- `maint-guide-ja`
- `debian-policy`
- `lintian`
- `developers-reference-ja`
- Debian Backports - Contribute <https://backports.debian.org/Contribute/>
- DEP-3: Patch Tagging Guidelines <http://dep.debian.net/deps/dep3/>
- 小林儀匡 “パッチ管理ツール quilt の使い方” 2007 年 1 月 第 24 回東京エリア Debian 勉強会資料
- 岩松信洋 “基本的なパッケージの作成方法手引書” 2013 年 2 月 Debian パッケージング道場資料
- 岩松信洋 “git build-package” 2015 年 9 月 第 130 回東京エリア Debian 勉強会 Debian パッケージング道場 資料



6 Debbugs とのつきあいかた:SOAP 編

林健太郎 (kenhys)

6.1 はじめに

Debian を使っていて不具合に遭遇したら、BTS(bugs.d.o) を参照して既知の不具合でないか確認することでしょう。

BTS の閲覧には、ブラウザを使うとか reportbug、あるいは reportbug-ng などを使います。このアクセス先の BTS が Debbugs です。今回は Debbugs とやりとりするための、もうひとつのやりかたについて紹介します。

6.2 Debbugs とは？

1994 年 Ian Jackson 氏が開発を始めた Issue トラッキングシステムです。GNOME や KDE もかつて採用していましたが、今では Bugzilla へ移行しました。

```
https://bugs.debian.org/cgi-bin/pkgreport.cgi?dist=unstable;package=debbugs

Debian Bug report logs: Bugs in package debbugs (version 2.4.1.1)
Maintainers for debbugs are Debbugs developers <debian-debbugs@lists.debian.org>.
You may want to refer to the following packages that are part of the same source: debbugs-local, debbugs
You might like to refer to the debbugs package page, to the Package Tracking System, or to the source
If you find a bug not listed here, please report it.

■ Control \(service\) bugs (25 bugs)
■ nnn@ \(process\) bugs (24 bugs)
■ Bug CGI bugs (21 bugs)
■ Package CGI bugs (28 bugs)
■ Documentation bugs (5 bugs)
■ SOAP/REST API bugs (11 bugs)
■ Uncategorized bugs (20 bugs)
■ Misc bugs (37 bugs)
■ Packaging bugs (2 bugs)
■ Fixed in Git bugs (49 bugs)
■ Wontfix bugs (16 bugs)
■ Fixed bugs (6 bugs)

Control (service) bugs (25 bugs)

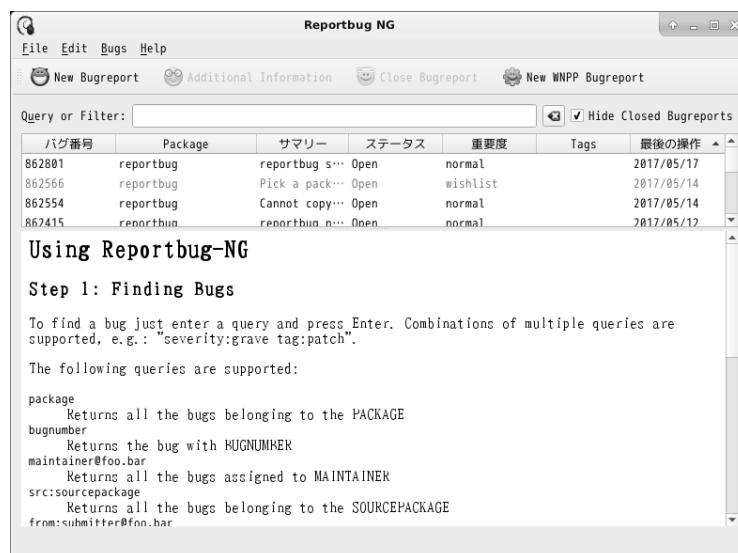
■ #49228 [w] [!] [debbugs] abstract out control@ interface and allow more use of Pseudoheaders:
■ #364883 [m] [ ] [debbugs] [service.in] reassign doesn't work with multiple packages+versions.
■ #376293 [w] [ ] [debbugs] debbugs: Add chaining options to "block" command (when A is closed, it clo
```

Debbugs のスクリーンショットがこれです。おなじみのやつですね。

バグ報告ではおなじみ、reportbug <https://packages.debian.org/ja/stretch/reportbug> がこちらです。対話的にバグ報告をすすめることができます。



他に、「Debian の古典的な reportbug の使いやすい置き換え」を目指している reportbug ng <https://packages.debian.org/ja/stretch/reportbug-ng> というものもあります。



6.3 Debbugs のよいところ

Debbugs にはいくつかよいところがあります。

- メールで気軽にバグ報告できる
- ログインとか不要で報告できる
- 一見さんをカジュアルに殺せる初見殺し機能^{*30}を搭載している

6.4 Debbugs の見方

バグ一覧をパッとみたときに、それが一体どのような意味を持つかは慣れていないとなかなかわかりにくいかもしれません。以下にその例を示します。

*30 まず何をどうしたらいいのかわからない

Control (service) bugs

```
■ #49228 [w|_|=] [debbugs] abstra
■ #364803 [m|_|_] [debbugs] [ser
■ #376293 [w|_|_] [debbugs] debb
- #422120 [m|_|_] [debbugs] BPS
```

左から順に、「バグ番号、重要度、タグ、状態、パッケージ名、件名」を示します。

Debbugs の重要度には次のものがあります。

- m: minor
- w: wishlist
- n: normal
- S: serious
- G: grave

タグには次のものがあります。

- +: patch
- ☹: wontfix
- M: moreinfo
- P: pending

状態には次のものがあります。

- ☹: blocks
- ☹: affects
- ☺: fixed

最後の状態などは、言われてみればなるほど、と思うかもしれません。

6.5 なぜ Debbugs に興味を?

冒頭に書いたように、なぜあえてもうひとつのやり方を探す必要があったのでしょうか。これには次のような理由がありました。

社内のグループチャットが Zulip <https://zulip.org/> であり、Zulip の bot に Debbugs をサポートさせたいというのがそもそもの動機でした。バグ報告の URL を貼り付けたら bot にいい感じに処理させたい、という要求があったためです。^{*31}

さて、バグ報告内容を bot に処理させるにはどうするのがよいでしょうか。bugs.d.o をスクレイピングするというのが安直ですが、もっといいやり方があります。

そこで見つけたのが <https://wiki.debian.org/DebbugsSoapInterface> です。実は Debbugs には SOAP のインターフェースがあったのですね。

該当ページにはもともと Ruby のサンプルコードもありました。

```
require 'soap/rpc/driver'

host = "bugs.debian.org"
port = 80
server="http://#{host}:#{port}/cgi-bin/soap.cgi"
ns = 'Debbugs/SOAP/'
drv = SOAP::RPC::Driver.new(server, ns)
drv.wiredump_dev = STDOUT if $DEBUG
drv.add_method('get_status', 'bugnumber')
drv.add_method('get_bugs', 'keyparam')

p drv.get_status(drv.get_bugs(['package', 'pbuilder', 'severity', 'wishlist']))
```

例えば上記のようなコードです。ただし、Ruby 1.8 or 修正済み soap4r じゃないと動かないという罠がありました。

^{*31} 結局はその機能は不要になりましたが。。

修正済み soap4r とあるように、実は soap4r には微妙な事情がありました。

- soap4r はもともと Ruby 1.8 にバンドル
- Ruby 1.9 ではバンドルされなくなった
- soap4r はメンテされなくなった
- 雨後の筍のように soap4r の派生ができた

ここでどれくらい雨後の筍感があるのか見てみましょう。

- soap4r-ng (2.0.3)
- soap4r-r19 (1.5.9) Ruby1.9 対応版
- soap4r-ruby1.9 (2.0.5) Ruby1.9 対応版
- soap4r-ruby19 (1.5.9) Ruby1.9 対応版

なかなか混沌としています。しかしことはそれだけでは終わりません。

soap4r の派生の soapXr というバージョンもありました。

- soap2r (1.5.8) soap4r の後にリリース
- soap5r (2.0.3)

Debian の場合は ruby-soap4r <https://github.com/noeticpenguin/soap4r-noeticpenguin> が upstream とされています。がそのあたりの事情はよくわかりません。あまりにも混沌とした世界なので、2017 年にもなって、あまり踏み込まないほうがよい予感がしてきます。

6.6 SOAP を使うには

ひとまず Ruby な事情は置いておくとして、SOAP インターフェースを使うときのことを考えてみましょう。

このときポイントとなるキーワードが WSDL です。WSDL とは「Web Services Description Language」のことで、いわゆる Web サービス記述言語です。XML でインターフェースを定義し、WSDL を元に SOAP で通信します。

では、Debbugs の WSDL とはどこにあるのでしょうか。 <https://wiki.debian.org/DebbugsSoapInterface> には一切言及はありませんでした。どうなってるのでしょうか？^{*32}

WSDL を探してみるとありました。



git index : emacs/elpa.git
ELPA

summary refs log tree commit diff

path: root/packages/debbugs

Mode	Name
-rw-r--r--	Debbugs.wsd1
-rw-r--r--	README
-rw-r--r--	debbugs-browse.el
-rw-r--r--	debbugs-gnu.el
-rw-r--r--	debbugs-org.el
-rw-r--r--	debbugs-ug.info
-rw-r--r--	debbugs-ug.texti
-rw-r--r--	debbugs.el
-rw-r--r--	debbugs.info
-rw-r--r--	debbugs.texti
-rw-r--r--	dir

どうやら Emacs 使いは debbugs.el <https://elpa.gnu.org/packages/debbugs.html> を使っているようです。Debbugs.wsd1 というのがそれです。Debbugs 用の WSDL で、debian.org と gnu.org の Debbugs で使えるもののようです。

^{*32} 2017 年 7 月現在、調査結果を追記済みです。

6.7 WSDL の構造

ここで、WSDL の構造についてもおさらいしておきましょう。すべての要素は <wsdl:definitions> の子要素として定義します。

- <wsdl:types>
- <wsdl:message>
- <wsdl:portType>
- <wsdl:binding>
- <wsdl:service>

6.7.1 <wsdl:types>

```
<complexType name="ArrayOfBugNumber">
  <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:int []"/>
    </restriction>
  </complexContent>
</complexType>
```

データ型の定義をするのに使います。上記の例はバグ番号の配列を定義しています。

6.7.2 <wsdl:message>

```
<wsdl:message name="get_statusRequest">
  <wsdl:part name="bugs" type="types:ArrayOfBugNumber"/>
</wsdl:message>
<wsdl:message name="get_statusResponse">
  <wsdl:part name="s-gensym3" type="apachens:Map"/>
</wsdl:message>
```

データの抽象的な定義をするのに使います。上記の例はステータス取得時にやりとりするデータの定義です。

6.7.3 <wsdl:portType>

```
<wsdl:portType name="Debbugs/SOAP">
  <wsdl:operation name="get_status" parameterOrder="bugs">
    ...
  </wsdl:operation>
  <wsdl:operation name="get_bug_log" parameterOrder="bugnumber">
    ...
  </wsdl:operation>
</wsdl:portType>
```

操作の定義に使います。<wsdl:operation> を含みます。

6.7.4 <wsdl:operation>

```
<wsdl:operation name="get_status" parameterOrder="bugs">
  <wsdl:input message="tns:get_statusRequest">
    <soap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:Debbugs/SOAP"
      use="encoded"/>
  </wsdl:input>
  <wsdl:output message="tns:get_statusResponse">
    <soap:body
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="urn:Debbugs/SOAP"
      use="encoded"/>
  </wsdl:output>
</wsdl:operation>
```

操作の入出力を定義するのに使います。上記の例はステータス取得時の入出力を定義しています。

6.7.5 <wsdl:binding>

```
<wsdl:binding name="Debbugs/SOAP/BINDING" type="tns:Debbugs/SOAP">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="get_status">
    ...
  </wsdl:operation>
</wsdl:binding>
```

portType で定義した操作のプロトコルを指定します。

6.7.6 <wsdl:service>

```
<wsdl:service name="Debbugs/SOAP/SERVICE">
  <wsdl:port binding="tns:Debbugs/SOAP/BINDING" name="gnu.org">
    <wsdlsoap:address location="http://debbugs.gnu.org/cgi/soap.cgi"/>
  </wsdl:port>
  <wsdl:port binding="tns:Debbugs/SOAP/BINDING" name="debian.org">
    <wsdlsoap:address location="http://bugs.debian.org/cgi-bin/soap.cgi"/>
  </wsdl:port>
</wsdl:service>
```

通信先を定義するのに使います。

6.8 実践:Debbugs

では、実際に SOAP でやりとりしてみましょう。Ruby の SOAP クライアントライブラリーの savon <http://savonrb.com/> を使います。

require "savon" してから、WSDL からクライアントのインスタンスを作成します。次に call で Debbugs のメソッドを呼び、レスポンスを表示すれば OK です。

```
client = Savon.client(wsdl: "(WSDL の URL)",
  endpoint: "http://bugs.debian.org/cgi-bin/soap.cgi",
  namespace: "Debbugs/SOAP")
```

インスタンスは上記のようにして取得します。

```
\# fetch status of pbuilder specific bugs
response = client.call(:get_status) do
  message(bugs: ["807406", "837812"])
end
```

上記は get_status メソッドを呼ぶ例です。

```
bugs = response.body[:get_status_response][:s_gensym3][:item]
bugs.each do |bug|
  item = bug[:value]
  puts "#{bug[:key]}:#{item[:pending]}:#{item[:subject]}"
end
\#=>
807406:pending:pbuilder: drop all the 'xenial' (...)
837812:pending:[pbuilder] New hook improving dpkgi
```

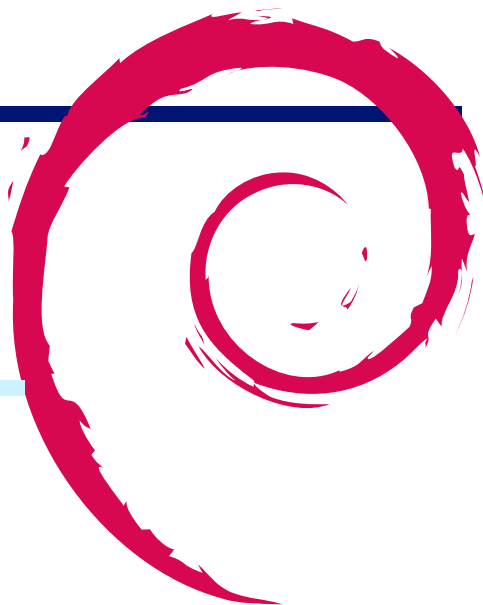
あとはレスポンスから必要な情報を取りだすだけです。上記はバグ番号や、状態、件名を表示しています。

6.9 まとめ

- debbugs には SOAP インターフェースがある
- Debbug.wsdl は debbugs.el 方面から入手せよ
- Ruby から使うなら soap4r(修正版) or savon を使え

7 Debian Continuous Integration

Charles Plessy



7.1 Introduction

Debian Continuous Integration (debc) is both a running service (<https://ci.debian.net/>) and a set of packages powering the service, (<https://packages.debian.org/sid/debc>). The website is mobile-friendly, and cross-links with the Debian Package Tracker.

Debian runs debci on `unstable`. I want to run Debci on a local version of `Stretch` because a) a late update of `r-base` unexpectedly broke API and b) a further upload of `r-base` in `unstable` means that `ci.debian.net` does not report a situation reflecting the state of Squeeze.

(Very embarrassingly, I have to confess during this release party... some packages might still be broken at the moment...)

First, let's see what is Debci and how it works.

7.2 Regression tests at build time

Traditionally, in Debian regression tests have been running at build time.

Advantage: they run on all architectures.

Limitations:

- They run only once in the lifetime of the package.
- Their output is deep in the build logs.
- For a long time, they did not run for architecture-independent packages, because they were not autobuild. (This changed recently with the possibility of source-only uploads).
- Obviously, they require to build the package first, which may take time and require extra packages (for instance to build the documentation, etc...).

What if a binary package gets broken by the update of one of its dependencies ?

7.3 Continuous Integration

Advantages:

- Ran again and again each time a package's environment changes.
- No need to build from source.
- Opportunity to test features not available at build time (network, reboot, ...)

Limitations:

- At the moment, Debian only runs them on `amd64` and `arm64`.

Debian Continuous Integration uses the Autopkgtest framework.

7.4 Autopkgtest

Also known as *DEP 8* (<http://dep.debian.net/deps/dep8/>). In brief:

The presence of a test suite is announced by the line `Testsuite: autopkgtest` in the Debian Source Control file.

Regression tests are described in `debian/tests/control` in the *paragraph* (“pseudo-RFC-822”) format well known to Debian packagers. The description is mostly about where to find the tests, and what they need to run.

When there are already run-time regression tests available upstream, populating `debian/tests` is easy. Example of the package `r-bioc-s4vectors`:

```
[r-bioc-s4vectors-0.12.1]$ cat debian/tests/control
Tests: run-unit-test
Depends: @, r-cran-runit, r-bioc-iranges (>= 2.0.0), r-bioc-genomicranges
Restrictions: allow-stderr

[r-bioc-s4vectors-0.12.1]$ cat debian/tests/run-unit-test
#!/bin/sh -e

LC_ALL=C R --no-save <<EOT
require("S4Vectors")
S4Vectors:::.test()
EOT
```

There are at least two ways to run `Autopkgtest` suites.

- For occasional and simple use `sadt`, from the `devscripts` package.
- For heavier or more complex use, `autopkgtest`, from the `autopkgtest` package. (Formerly, the command was called `adt-run`). One of the main advantages over `sadt` is the isolation of the test environment using containers or virtualisation.

7.5 Containerization of the test bed

`autopkgtest` supports multiple virtualisation systems:

- `schroot`
- `LXC`
- `QEMU`
- `ssh` (for instance to connect to a disposable cloud instance).

There is also the `null` system for running tests directly on the machine, like with `sadt`.

I tried `schroot`, because I am most familiar with it as it is used in the Debian build farm, and I use `sbuild` to build my packages. I created a `schroot` with the command `sbuild-createchroot`, and modified the configuration according to <https://ci.debian.net/doc/file.MAINTAINERS.html>

```
$ cat /etc/schroot/chroot.d/debci-stretch-amd64
[debci-stretch-amd64]
description=Debian stretch/amd64 CI testbed
users=debci,charles
root-users=debci,charles
groups=root,sbuild
root-groups=root,sbuild
type=directory
directory=/srv/chroot/stretch-debci
union-type=overlay
```

In retrospect, I should have created it with `debci update-worker` (see below).

The test bed can then be used to run tests with `autopkgtest`. For example:

```
autopkgtest --user debci --output-dir /tmp/output-dir \
  r-bioc-biostrings -- schroot debci-stretch-amd64
```

A comprehensive output is kept after the tests are run.

```
ls /tmp/output-dir/
log run-unit-test-packages run-unit-test-stdout summary testbed-packages testinfo.json testpkg-version
```

```

$ head /tmp/output-dir/*
==> /tmp/output-dir/log <==
adt-run [06:58:08]: version 4.4
adt-run [06:58:08]: host bubu; command line: /usr/bin/adt-run --user debci --output-dir /tmp/output-dir r-bioc-bio
adt-run [06:58:08]: testbed dpkg architecture: amd64
adt-run [06:58:08]: testbed running kernel: Linux 4.9.0-2-amd64 #1 SMP Debian 4.9.18-1 (2017-03-30)
adt-run [06:58:08]: @@@@@@@@@@@@@@@@@@ apt-source r-bioc-biostrings
Get:1 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (dsc) [2228 B]
Get:2 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (tar) [12.7 MB]
Get:3 http://deb.debian.org/debian stretch/main r-bioc-biostrings 2.42.1-1 (diff) [4008 B]
adt-run [06:58:12]: testing package r-bioc-biostrings version 2.42.1-1
adt-run [06:58:12]: build not needed

==> /tmp/output-dir/run-unit-test-packages <==
ca-certificates 20161130+nmu1
fontconfig 2.11.0-6.7+b1
fontconfig-config 2.11.0-6.7
fonts-dejavu-core 2.37-1
libblas-common 3.7.0-2
libblas3 3.7.0-2
libcairo2 1.14.8-1
libcurl3 7.52.1-5
libdatrie1 0.2.10-4+b1
libexpat1 2.2.0-2

==> /tmp/output-dir/run-unit-test-stdout <==

R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.

==> /tmp/output-dir/summary <==
run-unit-test PASS

==> /tmp/output-dir/testbed-packages <==
adduser 3.115
apt 1.4.4
apt-utils 1.4.4
base-files 9.9
base-passwd 3.5.43
bash 4.4-5
binutils 2.28-5
bsdmainutils 9.0.12+nmu1
bsdutils 1:2.29.2-1
bzip2 1.0.6-8.1

==> /tmp/output-dir/testinfo.json <==
{
"virt_server": "autopkgtest-virt-schroot debci-stretch-amd64",

```

```

"cpu_model": "Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz",
"kernel_version": "Linux 4.9.0-2-amd64 #1 SMP Debian 4.9.18-1 (2017-03-30)",
"nproc": "4",
"cpu_flags": "fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
}
==> /tmp/output-dir/testpkg-version <==
r-bioc-biostrings 2.42.1-1

```

The *debc* system is there to manage a queue of tests dispatched to workers, and collect the results.

7.6 Running debci locally

Debc is made of four components: an injector, a message queue, workers and a collector.

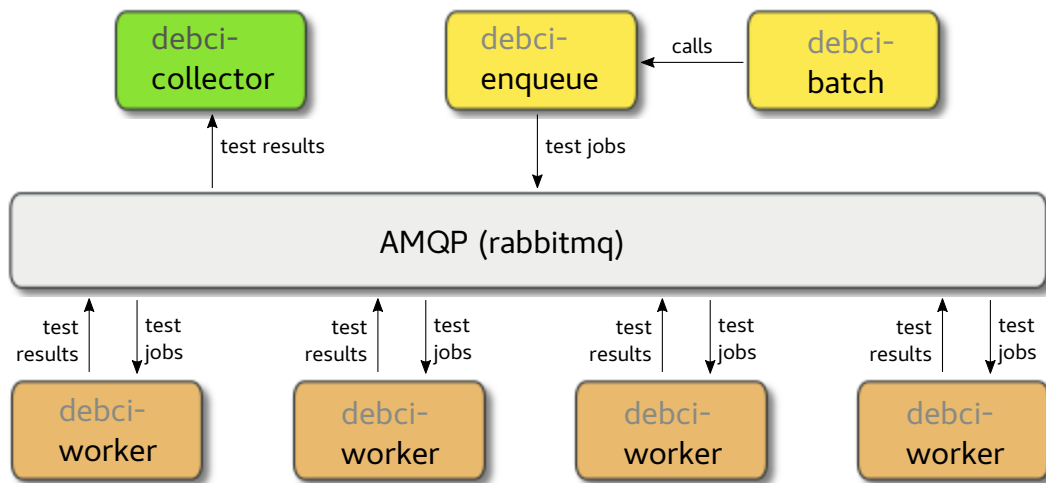


Figure 1: debci's architecture

Installation instructions: <https://ci.debian.net/doc/file.INSTALL.html>

```
sudo apt install debci debci-collector debci-worker
```

It is also recommended to use `apt-cacher-ng` (not sure if just installing is enough...).

7.6.1 Issues with the installation.

Disclaimer: *maybe everything works well and I did not forget the instructions correctly? I did not have time to triple-check.*

Debc's documentation says: "As usual, you will prompted for the address of the AMQP server. If the rabbitmq-server is on the same host, just leave it blank". In my case, I did not see a prompt. But since I use a local server, there was no problem.

If there are problems with rabbitmq, it is useful to check at the web interface. But it is not installed by default.

```
rabbitmq-plugins enable rabbitmq_management
```

Then, visit `http://localhost:15672/` (login:guest, password:guest0).

There is also a command-line tool to query the queues: `rabbitmqadmin`.

7.6.2 Other issues.

<https://bugs.debian.org/809652>: *debc: setup sets stamps even if setup failed*

This bug prevents debci commands to try again when they should. For example, this command fails for a good reason (lxc not installed).

```
$ sudo debci update-worker
```

```
Starting testbed setup: Tue May 30 21:17:47 JST 2017
```

```
E: lxc is not installed
```


Finished testbed setup: Tue May 30 21:17:47 JST 2017

What happens if one configure lxc as needed and restarts the command ?

```
$ sudo debci update-worker
```

Starting testbed setup: Tue May 30 21:17:50 JST 2017

```
I: testbed already updated in the last 12h, no need to update
```

It refuses to run for the next 12 hours !

Fortunately the solution is simple: there is a `--force` option.

The debci index generator was crashing and I could not debug the problem, but I happened to have to reboot my laptop this week, and the problem disappeared... (Unfortunately, `systemd` did not keep the logs of the previous boots... how did I configure it so wrong ?)

7.6.3 Configuration

`lxc` is default, but I wanted to use `schroot`. Either I had to pass `--backend schroot` to all the commands, or I had to add it to the configuration (same for `--suite stretch`).

The configuration files are in `/etc/debci/` and `/debci/conf.d/`. By default, these directories are empty, with no example files. But study of the source code of debci indicates that the configuration files are sourced by the shell to define environment variables. Further study of the debci code suggested that the names of the variables can easily be guessed. So here is my configuration.

```
$ cat /etc/debci/conf.d/local.conf
debci_suite=stretch
debci_backend=schroot
```

TODO: submit an example configuration file via the BTS...

7.6.4 Et c'est parti !

Let's test all R/Bioconductor packages in Squeeze.

```
apt-cache search r-bioc* | cut -f1 -d ' ' | xargs debci enqueue
```

And voil'a, debci did the rest.

```
$ ls /var/lib/debci/data/packages/stretch/amd64/r/ | more
r-bioc-affy
r-bioc-affyio
r-bioc-altcdfenvs
[ plus many more ... ]
r-bioc-metagenomeseq
r-bioc-multtest
r-bioc-phyloseq
```

7.7 Conclusion

I still have to dig in the results.

There seems to be some autogenerated HTML pages, but I still have to figure out how to get nice diffs like on `ci.debian.net`. Example file path:

```
file:///var/lib/debci/data/.html/packages/r/r-bioc-affy/stretch/amd64/index.html
```

Final conclusion: I hope to use `debci` more extensively during the next release cycle to test potential backports.

8 challenge: convert debian-policy doc from docbook to Sphinx

Hideki Yamane



proposal: move forward to modern environment

8.1 Why move away from docbook?

Adding i18n support to docbook xml is so complicated couldn't do that with old debiandoc-sgml, at least I want to merge Japanese translation to upstream It's harder to edit/read than other markup language (such as Markdown, AsciiDoc, reStructuredText) Less effort, more profit Lower barrier for newcomers

8.2 Why Sphinx?

Linux Kernel documents have already moved from docbook to Sphinx They've already checked markdown and asciidoc, but finally choosed reStructuredText and Sphinx see https://youtu.be/jY_C-ZOqMS0 I know upstream maintainer, so I can ask him about its feature via twitter at any time :-)

8.3 Sphinx?

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.

8.4 Debian Policy: output format

Debian Policy Manual

This manual describes the policy requirements for the Debian GNU/Linux distribution. This includes the structure and contents of the Debian archive, several design issues of the operating system, as well as technical requirements that each package must satisfy to be included in the distribution.

Authors: Ian Jackson, Christian Schwarz, David A. Morris

Maintainer: The Debian Policy group

Status: ready

Availability: Debian package `debian-policy`
Latest version:

- **English:** [HTML] [PDF] [PS] [plain text]

The latest SGML/XML source is available through the [Git](#) repository.

- **Web interface:** <https://anonscm.debian.org/cgiit/dbnpolicy/policy.git/>
- **VCS interface:** `git clone git://anonscm.debian.org/dbnpolicy/policy`

Sphinx supports: HTML, LaTeX (PDF), ePub, Plain Text, Texinfo, manual pages

8.5 convert docbook .xml to .rst via pandoc

It 's fairly easy! `sudo apt install pandoc` `pandoc -f docbook -t rst policy.xml -o policy.rst`

8.6 Pandoc issue

Generates footnote as static number, not autonumbered sed docbook generated footnote for each section, but pandoc puts only one huge footnote manual edit

8.7 Pandoc issue

Break Internal Hyperlink reference

```
- 'section\_title <#s-fhs>'__  
+ :ref:'s-fhs'  
- ''Format'' <#s-f-Format>'__  
+ :ref:'Format <s-f-Format>'__  
- 'Control files and their fields <#ch-controlfields>'__  
+ :doc:'Control files and their fields <ch-controlfields>'.
```

8.8 Pandoc issue

Lose information

```
<!ENTITY changesversion "1.8">  
The format described in this document is <literal>&changesversion;</literal>.  
The format described in this document is ''''.
```

8.9 Pandoc issue

```
Link tag is bit strange  
<\url{https://www.example.com/}>'__  
expected: <\url{https://www.example.com/}>'_  
(last _ is single)
```

8.10 Sphinx: strong point

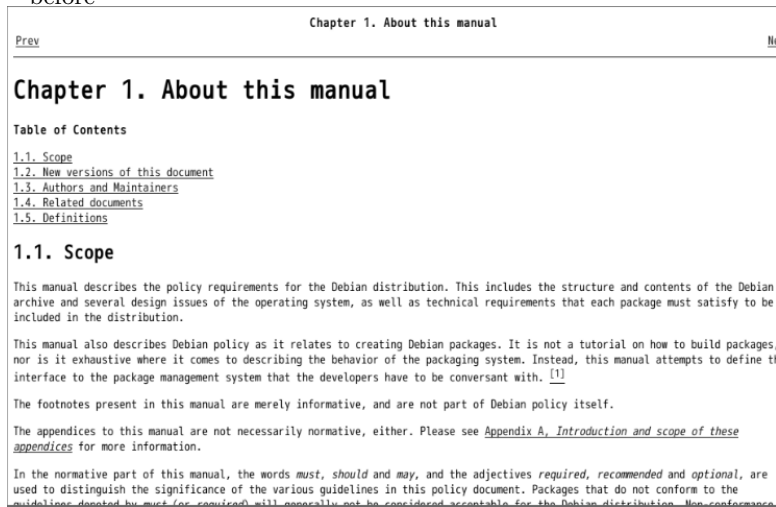
i18n: established way to add new language support

```
$ sudo apt install python-pip  
$ pip install sphinx-intl; export PATH=~/.local/bin:$PATH  
add locale setting to conf.py (edit)  
$ make gettext  
$ sphinx-intl update -p build/locale -l ja -l <lang2> ..  
translate source/locale/<lang>/LC_MESSAGES/<file>.po (edit)  
make html  
cd source && sphinx-build . ./build/html -D language='ja' -D html_file_suffix=.ja.html
```

see <http://www.sphinx-doc.org/en/stable/intl.html>

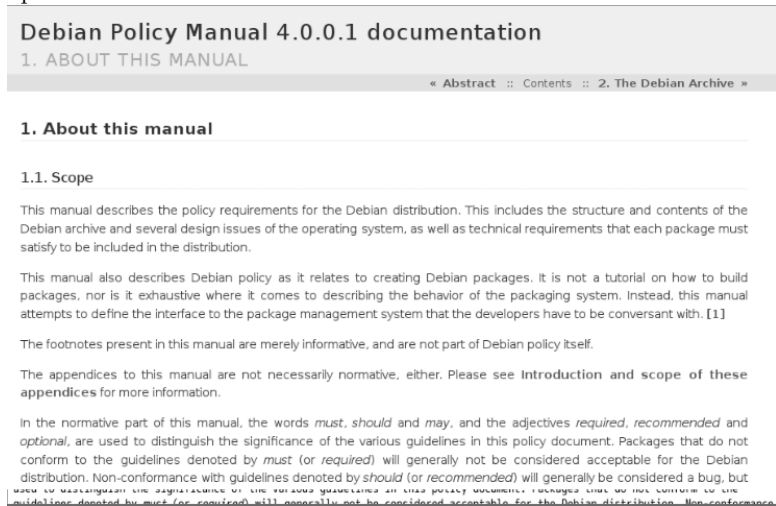
8.11 Sphinx: modern look

before



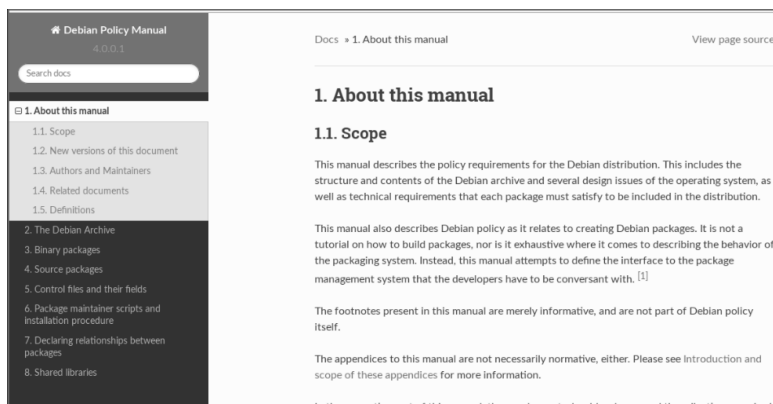
The screenshot shows a simple, text-based layout. At the top, it says "Chapter 1. About this manual" with "Prev" and "Next" links. Below is a "Table of Contents" with links to sections 1.1 through 1.5. Section 1.1, "Scope", is expanded, showing several paragraphs of text. The text is plain, with no styling, and includes footnotes and appendices information.

Sphinx



The screenshot shows a modern, styled Sphinx output. It features a header "Debian Policy Manual 4.0.0.1 documentation" and a navigation bar with "1. ABOUT THIS MANUAL" and "« Abstract :: Contents :: 2. The Debian Archive »". The main content area has a title "1. About this manual" and a sub-section "1.1. Scope". The text is styled with a clean, modern font and includes a search bar at the top left. The layout is more organized and visually appealing than the previous version.

8.12 Sphinx: “ theme ” feature



The screenshot shows the Sphinx output with a theme. It features a dark sidebar on the left with a search bar and a table of contents. The main content area has a title "1. About this manual" and a sub-section "1.1. Scope". The text is styled with a clean, modern font and includes a search bar at the top left. The layout is more organized and visually appealing than the previous version.

You can change HTML visual via “ jinja2 ” template (I 'm not sure...) see <http://www.sphinx-doc.org/en/stable/theming.html>

8.13 Limitation in Sphinx compared to docbook

No Appendix support footnote number does not continue for whole document Upstream kindly create & proposed extension! :) Cannot use nested tag :ref:“Maintainer“ <s-f-Maintainer>‘

8.14 Demo

```
$ sudo apt install python-sphinx python-pip
$ pip install sphinx-intl; export PATH=~/.local/bin:$PATH
$ git clone https://github.com/henrich/policydoc
$ cd policydoc
$ git checkout restructure
$ make -f sphinx-makefile html

$ cd source; sphinx-build ../build/html -D language='ja' -D html_file_suffix=.ja.html
```

8.15 Not yet

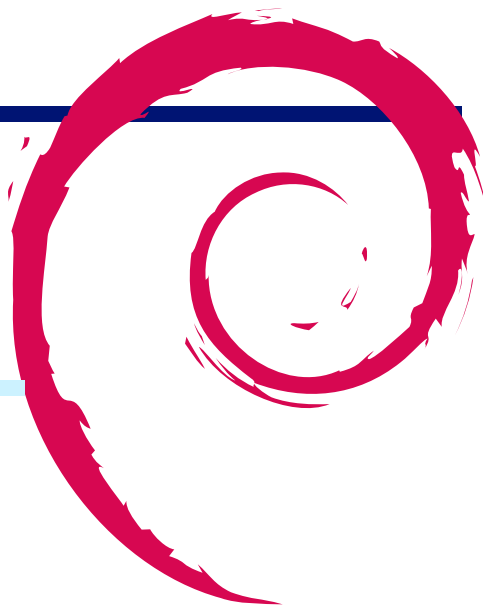
PDF output check, compare to docbook one *sudoptinstallltxmk;makelatexpdfePuboutputcheck* make epub ...
Maybe need to discuss sphinx upstream to improve it.

8.16 Conclusion Next step?

It 's (almost) ready to use sphinx for debian-policy documentation But needs improvement for support multi language output support Give a proposal to -policy mailing list ITP: sphinx-intl make “ debian doc ” theme for sphinx (needs help)

9 2016 年の振り返りと 2017 年の企画

Debian JP



2016 年も今月で最後の関西 Debian 勉強会になります。2007 年 3 月が初開催なので、来年 3 月で 10 周年です。

9.1 勉強会全体について

9.1.1 セッション、定例ネタ

2016 年前半のセッションは昨年末の企画に沿って「GNUHurd のインストールしてみた。と、X サーバの立ち上げに挑戦」、
「systemd に浸かってみる」、「OpenFOAM による数値流体解析」と発表していただきました。

他には、「VyOS を入れて AP を構築してみた。」、「周回遅れで Docker 触ってみた」、「Let 's Encrypt のススメ」、「sbuid
と debci を触ってみた」のセッション発表もしていただきました。

定例ネタとしては、「初心者が初めてパッケージをつくってみた」とパッケージング道場を行なうことができました。

9.2 運営

運営に関しては、毎月の継続した開催は行なってきました。

Doorkeeper の有料化により compass に移行しました。

昨年に続き、事前の準備、告知が不十分な場合がありますので、今後は、定型の告知を先に投げるなど、改善していきたいと考えています。

現運営だけではまわっていないところがありますので、取り纏め、告知などなど運営にご協力ください。

9.3 イベント/NM 申請

イベント参加については、OSC 2016 Kansai@Kyoto、KOF2016 に参加しました。今後も継続して参加する予定です。

Debian 開発者への道、としてここしばらく関西 Debian 勉強会からは動きがありません。

9.4 開催実績

関西 Debian 勉強会の出席状況を確認してみましょう。グラフで見ると図 2 になります。また、毎回の参加者、アンケート回答者の人数とその際のトピックを表 1 にまとめました。グラフ中の黒線は参加人数、赤線は 1 年の移動平均、青線はアンケートの回答人数です。参加人数が 0 となっているところは人数が集計されていない or 開催されなかった月です。アンケートの回答人数が 0 となっているところはアンケートが実施されなかった月です。

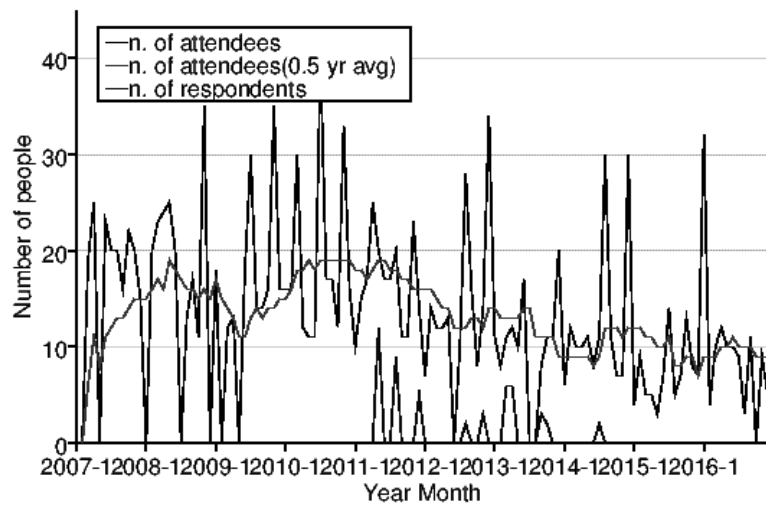


図 2 関西の参加人数推移 (参加人数と 6ヶ月移動平均、アンケート回答人数)

表 1 関西 Debian 勉強会の参加人数とトピック (2016)

開催年月	参加人数	内容
2016 年 1 月	12	GNUHurd のインストールしてみた。と、X サーバの立ち上げに挑戦 VyOS を入れて AP を構築してみた。
2016 年 2 月	10	勉強会資料の歩き方 周回遅れで Docker 触ってみた
2016 年 3 月	10	systemd に浸かってみる
2016 年 4 月	9	OpenFOAM による数値流体解析
2016 年 5 月	3	ライトニングトーク & もくもくの会
2016 年 6 月	11	Debian パッケージング道場
2016 年 7 月		OSC 2016 @ Kyoto
2016 年 8 月	9	ライトニングトーク & もくもくの会
2016 年 9 月	5	初心者が初めてパッケージをつくってみた Let's Encrypt のススメ
2016 年 10 月	7	sbuid と dehci を触ってみた
2016 年 11 月		KOF 2016
2016 年 12 月	7	2016 年の振り返りと 2017 年の企画, 忘年会

表 2 関西 Debian 勉強会の参加人数とトピック (2007-2008)

開催年月	参加人数	内容
2007年3月	19	開催にあたり
2007年4月	25	goodbye、youtube、プロジェクトトラッカー
2007年6月	23	社会契約、テーマ、debian/rules、bugreport
2007年7月	20 前後	OSC-Kansai
2007年8月	20	Inkscape、patch、dpatch
2007年9月	16	ライブラリ、翻訳、debtorrent
2007年10月	22	日本語入力、SPAM フィルタ
2007年11月	20 前後	KOF
2007年12月	15	忘年会、iPod touch
開催年月	参加人数	内容
2008年2月	20	PC Cluster, GIS, TeX
2008年3月	23	bug report, developer corner, GPG
2008年4月	24	coLinux, Debian GNU/kFreeBSD, sid
2008年5月	25	ipv6, emacs, us-tream.tv
2008年6月	20	pbuilder, hotplug, ssl
2008年8月	13	coLinux
2008年9月	17	debian mentors, ubiquity, DFSG
2008年10月	11	cdbs,cdn.debian.or.jp
2008年11月	35	KOF
2008年12月	?	TeX 資料作成ハンズオン

表 3 関西 Debian 勉強会の参加人数とトピック (2009-2010)

開催年月	参加人数	内容
2009年1月	18	DMCK, LT
2009年3月	12	Git
2009年4月	13	Installing sid, Mancoosi, keysign
2009年6月	18	Debian Live, bash
2009年7月	30?	OSC2009Kansai
2009年8月	14	DDTSS, lintian
2009年9月	14	reportbug, debian mentors
2009年10月	16	gdb, packaging
2009年11月	35	KOF2009
2009年12月	16	GPS program, OpenStreetMap
開催年月	参加人数	内容
2010年1月	16	Xen, 2010年企画
2010年2月	16	レンタルサーバでの利用, GAE
2010年3月	30?	OSC2010Kobe
2010年4月	12	デスクトップ環境, 正規表現
2010年5月	11	ubuntu, squeeze
2010年6月	11	debhelper7, cdbs, puppet
2010年7月	40?	OSC2010Kyoto
2010年8月	17	emdebian, kFreeBSD
pp 2010年9月	17	タイトル WM
2010年10月	12	initramfs, debian live
2010年11月	33	KOF2010
2010年12月	14	Proxmox, annual review

表 4 関西 Debian 勉強会の参加人数とトピック (2011-2012)

開催年月	参加	回答	内容
2011年1月	10	0	BTS, kFreeBSD
2011年2月	15	0	pbuilder, Squeeze リリースパーティ
2011年3月	17	0	ライセンス, ドキュメント
2011年4月	25	0	OSC Kansai@Kobe
2011年5月	20	12	vi, dpkg
2011年6月	17	0	vcs-buildpackage{svn, git}, IPv6
2011年7月	17	0	OSC Kansai@Kyoto
2011年8月	20	9	パッケージ作成ハンズオン
2011年9月	11	0	vcs-buildpackage{bzd, git}
2011年10月	11	0	Emacs, vim 拡張の Debian パッケージ, 翻訳
2011年11月	23	0	KOF 2011
2011年12月	13	5	NM プロセス, BTS
開催年月	参加	回答	内容
2012年1月	7	0	Debian 温泉合宿
2012年2月	14	0	autofs+pam_chroot, t-code, Debian Policy
2012年3月	12	0	Konoha, t-code, Debian Policy
2012年4月	12	0	フリーソフトウェアと著作権, Konoha, Debian Policy
2012年5月	13	0	Debian と LDAP(頓挫), ITP 入門, Debian Policy
2012年6月	-	0	大統一 Debian 勉強会
2012年7月	10	0	Debian と LDAP, Debian Policy
2012年8月	28	2	OSC Kansai@Kyoto
2012年8月	16	0	Debian と Kerberos, News from EDOS
2012年9月	8	0	clang, Debian Policy
2012年10月	14	3	翻訳環境構築, DSA の舞台裏
2012年11月	34	0	KOF 2012
2012年12月	12	0	Debian on Android, Debian Policy

表 5 関西 Debian 勉強会の参加人数とトピック (2013-14)

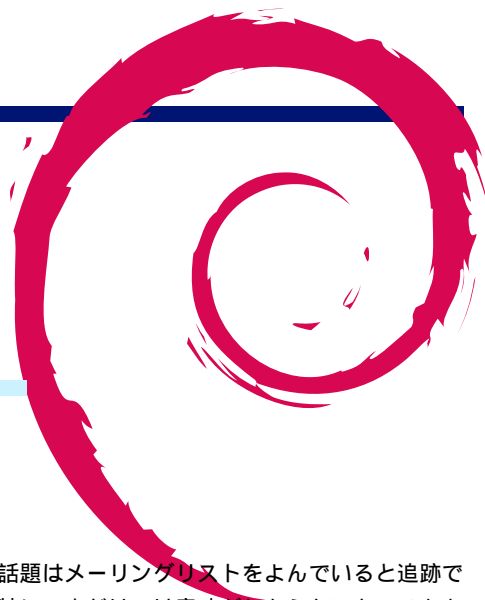
開催年月	参加	回答	内容
2013年1月	8	0	Drupal, Debian Policy
2013年2月	11	6	Debian Installer, Ruby
2013年3月	12	6	GNOME Shell, GNOME
2013年4月	10	0	リリースノート, AWS
2013年5月	17	0	Debian と Ubuntu, Debian の歩き方
2013年6月	-	0	大統一 Debian 勉強会
2013年7月		0	OSC 2013 Kansai @ Kyoto
2013年8月	8	3	puppet
2013年9月	11	2	サウンドシステム, dgit
2013年10月	11	0	ALSA, git-buildpackage
2013年11月	20	0	KOF 2013
2013年12月	6	0	2013年の振り返りと2014年の企画, 忘年会
開催年月	参加	回答	内容
2014年1月	12	0	LT, もくもくの会
2014年2月	10	0	もくもくの会
2014年3月	10	0	3D プリンティング, もくもくの会
2014年4月	11	0	kvm, Notmuch Mail
2014年5月	8	0	もくもくの会
2014年6月	11	2	systemd, Linux のドライバメンテナ, キーサイン
2014年8月	30	0	OSC 2014 Kansai @ Kyoto
	11	0	もくもくの会
2014年9月	7	0	もくもくの会
2014年10月	7	0	もくもくの会
2014年11月	30	0	KOF 2014
	4	0	インストララテスト, もくもくの会
2014年12月	9	0	2014年の振り返りと2015年の企画, 忘年会

表 6 関西 Debian 勉強会の参加人数とトピック (2015)

開催年月	参加人数	内容
2015 年 1 月	5	Debian Bug Squashing Party
	5	Debian の Bug の眺め方, Bug Squash Party
2015 年 2 月	3	もくもくの会
2015 年 3 月	7	某所 VPS を先走って Jessie に上げてみた
2015 年 4 月	14	Debian 8 "Jessie" Release Party
2015 年 5 月	5	Jessie 落穂拾い
2015 年 6 月	7	Wheezy Jessie で踏み抜いた地雷のご紹介
2015 年 7 月	13	OSC 2015 Kansai @ Kyoto
2015 年 8 月	8	wiki:Subkeys
2015 年 9 月	7	ドイツ、ハイデルベルクで開催された Debconf15 へいってきました
2015 年 11 月	32	KOF 2015
	4	ライトニングトーク, もくもくの会
2015 年 12 月	9	2015 年の振り返りと 2016 年の企画, 忘年会

10 Debian Trivia Quiz

DebianJP



ところで、みなさん Debian 関連の話題においついていますか？ Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではりあいが無いので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容と Debian Project News からです。

問題 1. Debian 9 (stretch) が 2017-01-07 に soft freeze に入ったことがアナウンスされました。soft freeze になると何ができなくなるでしょうか。

- A Debian Wiki の stretch ページの変更
- B 新規のソースパッケージのアップロード
- C 既存のソースパッケージのアップロード

問題 2. 2017 年の Debian Project Leader(=DPL) 選挙が行われ、投票結果が発表されました。DPL に選ばれたのは誰でしょうか。

- A Mehdi Dogguy
- B Nobuhiro Iwamatsu
- C Chris Lamb

問題 3. debian プロジェクトのパッケージ関連のサーバのうち、FTP サーバの提供を一部廃止することが発表されました。FTP サーバのうち、その後も継続稼働する予定のサーバはどれでしょうか。

- A `ftp://ftp.debian.org`
- B `ftp://security.debian.org`
- C `ftp://ftp.upload.debian.org`

問題 4. ついにリリースされた Debian 9 (Stretch)。リリースノートに書かれているリリースされたパッケージ数はおよそどのくらいでしょうか。

- A 40,000 パッケージくらい
- B 50,000 パッケージくらい
- C 60,000 パッケージくらい

問題 5. まだ話は早いですが、これからは次の Debian 10 に向かって作業を進めることとなります。Debian 10 のコードネームはなんでしょうか。

- A Potato
- B Bullseye
- C Buster

本資料のライセンスについて

本資料はフリー・ソフトウェアです。あなたは、Free Software Foundation が公表した GNU GENERAL PUBLIC LICENSE の "バージョン 2" もしくはそれ以降が定める条項に従って本プログラムを再頒布または変更することができます。

本プログラムは有用とは思いますが、頒布にあたっては、市場性及び特定目的適合性についての暗黙の保証を含めて、いかなる保証も行いません。詳細については GNU GENERAL PUBLIC LICENSE をお読みください。

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- If the modified program normally reads commands interactively when run, you must cause it, when started running for such

interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to

refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED

TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

ソースコードについて

このプログラムは `tex` で記述されたものです。ソースコードは

`git://anonscm.debian.org/tokyodebian/monthly-report.git`

から取得できます。

Debian オープンユースロゴ ライセンス

Copyright (c) 1999 Software in the Public Interest
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

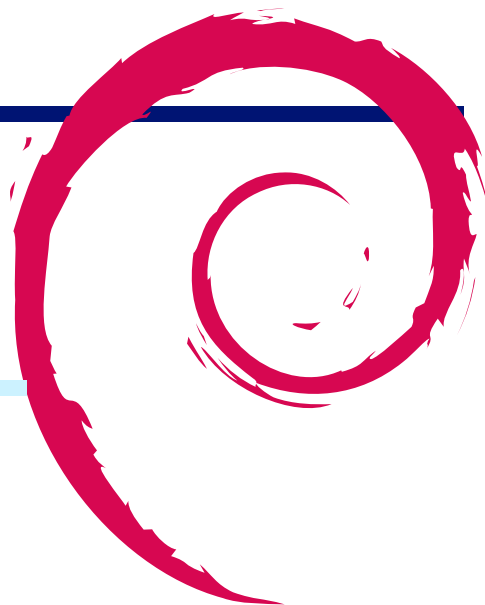
The above copyright notice and this permission notice shall be

included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

11 Debian Trivia Quiz 問題回答

DebianJP



Debian Trivia Quiz の問題回答です。 あなたは何問わかりましたか？

1. B soft freeze に入ると既存パッケージの品質向上に注力せよ、ということで新規のソースパッケージをアップロードできなくなります。full freeze は 2017-02-05 に予定されています。パッケージが unstable から testing へ移行するまで最短 10 日間かかるため、早めに修正してアップロードしましょう。 <https://lists.debian.org/debian-devel-announce/2017/01/msg00002.html>
2. C Chris Lamb さんが選出されました。おめでとうございます! <https://bits.debian.org/2017/04/results-dpl-elections-2017.html>。なお、2017 年 DPL 選挙の情報は次の web ページを参照ください https://www.debian.org/vote/2017/vote_001。
3. C 「Shutting down public FTP services」という表題でアナウンスされました。apt-line に指定するプロトコルは停止予定の 2017-11-01 までに”http://”に変更しましょう。なお、開発者向けの”ftp://ftp.upload.debian.org”、”ftp://security-master.debian.org”はその後も継続稼働するとのことです。 <https://lists.debian.org/debian-announce/2017/msg00001.html>。
4. B リリースノートの「2.2. ディストリビューションの最新情報」(<https://www.debian.org/releases/stretch/amd64/release-notes/ch-whats-new.html>) の記載によると、51687 パッケージを超えるとのことです (補足: main のバイナリパッケージの数です)。DD、DM、PM のみなさま、作業ありがとうございました。
5. C Debian 10 のコードネームは Buster になります。Debian のリリース情報は「<https://wiki.debian.org/DebianReleases>」にまとまっていますのでご参照ください。

『あんどきゅめんてっど でびあん』について

本書は、東京および関西周辺で毎月行なわれている『東京エリア Debian 勉強会』および『関西 Debian 勉強会』（2016年12月-2017年7月 OSC 2017 Hokkaido 出張版）で使用された資料・小ネタ・必殺技などを一冊にまとめたものです。内容は無保証、つっこみなどがあれば勉強会にて。



あんどきゅめんてっど でびあん 2017年夏号

2017年8月11日 初版第1刷発行

東京エリア Debian 勉強会/関西エリア Debian 勉強会（編集・印刷・発行）
